Software Requirements Specification

for

<Project>

Version 1.0 approved

Prepared by <author>

<organization> <date created>

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Software Requirements Specification for <Project> Page ii

# Table of Contents

[Table of Contents ii](#_Toc6145)

[Revision History ii](#_Toc6146)

[1. Introduction 1](#_Toc6147)

[1.1 Purpose 1](#_Toc6148)

[1.2 Document Conventions 1](#_Toc6149)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc6150)

[1.4 Product Scope 1](#_Toc6151)

[1.5 References 1](#_Toc6152)

[2. Overall Description 2](#_Toc6153)

[2.1 Product Perspective 2](#_Toc6154)

[2.2 Product Functions 2](#_Toc6155)

[2.3 User Classes and Characteristics 2](#_Toc6156)

[2.4 Operating Environment 2](#_Toc6157)

[2.5 Design and Implementation Constraints 2](#_Toc6158)

[2.6 User Documentation 2](#_Toc6159)

[2.7 Assumptions and Dependencies 3](#_Toc6160)

[3. External Interface Requirements 3](#_Toc6161)

[3.1 User Interfaces 3](#_Toc6162)

[3.2 Hardware Interfaces 3](#_Toc6163)

[3.3 Software Interfaces 3](#_Toc6164)

[3.4 Communications Interfaces 3](#_Toc6165)

[4. System Features 4](#_Toc6166)

[4.1 System Feature 1 4](#_Toc6167)

[4.2 System Feature 2 (and so on) 4](#_Toc6168)

[5. Other Nonfunctional Requirements 4](#_Toc6169)

[5.1 Performance Requirements 4](#_Toc6170)

[5.2 Safety Requirements 5](#_Toc6171)

[5.3 Security Requirements 5](#_Toc6172)

[5.4 Software Quality Attributes 5](#_Toc6173)

[5.5 Business Rules 5](#_Toc6174)

[6. Other Requirements 5](#_Toc6175)

[Appendix A: Glossary 5](#_Toc6176)

[Appendix B: Analysis Models 5](#_Toc6177)

[Appendix C: To Be Determined List 6](#_Toc6178)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason For Changes | Version |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this document is to define the software requirements for the **College Chatbot using Retrieval-Augmented Generation (RAG)**. This chatbot will help students, faculty, and visitors by answering queries related to admissions, courses, faculty, exams, placements, and campus facilities. The SRS covers all necessary details for building the system and outlines the development requirements.

## Document Conventions

The document follows standard conventions for software requirements specifications. Key terms and concepts are highlighted in bold or italics. Functional requirements are itemized and clearly labeled with unique identifiers like REQ-1.

## Intended Audience and Reading Suggestions

This app is primarily for people who want to get information about KMIT and also intended for students who are unaware of information regarding college. This webapp is also for faculty who want to know information regarding college.

## Product Scope

The product provides a way to answer college related queries using RAG approach. It will be available via web app interface. It can be accessible for users regarding courses,exams,admissions,fees.

## References

<https://medium.com/credera-engineering/build-a-simple-rag-chatbot-with-langchain-b96b233e1b2a>  
<https://arxiv.org/abs/2005.11401>

# Overall Description

## Product Perspective

This project will build a chatbot using RAG, a combination of retrieval-based search and text generation, to assist with answering college-related queries. The system will function as a web service accessible to students, faculty, and other visitors.

## Product Functions

**Data Preprocessing**: Clean and organize data from websites like KMIT official website and scrape the data from it.

**Embedding Model**: Use BERT/SBERT to generate embeddings and store them in a vector database.

**Retrieval Model**: Retrieve relevant documents based on user queries using the vector database.

**Generation Model**: Generate responses using a transformer-based model.

**Frontend Integration**: Provide a user-friendly interface for interaction with the chatbot.

## User Classes and Characteristics

**Students**: Frequently interact with the chatbot to ask about courses, exams, and campus events.

**Faculty**: Infrequent use for administrative purposes, querying details on courses or exam schedules.

**Visitors**: Occasional interactions, mainly related to admission information or campus facilities.

## Operating Environment

**Hardware**: The software will run on standard server hardware.

**Software**: The chatbot will be hosted on cloud platforms like AWS.

## Design and Implementation Constraints

The system should adhere to the college's data privacy policies.

The chatbot’s response time should be under 2 seconds.

Compatibility with existing college websites and infrastructure is required.

## User Documentation

User Manual: Instructions for interacting with the chatbot.

API Documentation: For developers integrating the chatbot into other systems.

## Assumptions and Dependencies

Assumes access to a large dataset of college-related documents for training the chatbot. Website information is not enough. Depends on cloud infrastructure (AWS) for deployment.

# External Interface Requirements

## User Interfaces

The chatbot will be accessible through a web-based user interface, designed using **React** for mobile-friendly interactions. The UI will allow users (students, faculty, visitors) to input their queries.

## Hardware Interfaces

The system is designed to be hosted on cloud infrastructure (AWS). The hardware components will include:

**Server instances** for running the backend services and models.

**GPUs** (on cloud) for efficient processing of transformer models.

## Software Interfaces

The product will integrate with various external software components, including:

**Operating System**: Linux-based servers on AWS/GCP/Azure.

**Frontend UI**: React

**Vector Database**: FAISS for fast retrival

## Communications Interfaces

The system will use the following communication methods:

**Message Formatting**: Responses will be sent as JSON, with the key-value pairs indicating the response text and status codes.

**Security**: Communication will be secured using SSL/TLS encryption to protect user data and ensure secure transmission between the client and server.

# System Features

## Data Preprocessing

### Description and Priority

This feature processes various college-related documents, such as FAQs, syllabi, notices, and more, to make them usable for the chatbot.

### Stimulus/Response Sequences

**Stimulus**: The user submits a query to the chatbot.

**Response**: The chatbot processes the query, retrieves relevant documents, and provides an appropriate response based on those documents.

### Functional Requirements

REQ-1: Extract text from documents (e.g., PDFs, Word files).

REQ-2: Clean and preprocess text data (remove unnecessary content, special characters, etc.).

REQ-3: Segment large documents into smaller, manageable chunks for embedding.

REQ-4: Store processed data in a suitable format for retrieval

## Embedding and Retrieval Models

### Description and Priority

This feature involves converting text documents into vector embeddings using **BERT** or **SBERT** and storing them in a vector database (FAISS)

**4.2.2 Stimulus/Response Sequences**

* **Stimulus**: The user submits a query to the chatbot.
* **Response**: The system retrieves the top-ranked documents using vector embeddings and feeds them to the generation model.

**4.2.3 Functional Requirements**

REQ-1: Use **BERT/SBERT** to generate document embeddings from preprocessed data.

REQ-2: Store embeddings in **FAISS** for efficient retrieval.

REQ-3: Implement a query processing mechanism that compares user input with stored embeddings for similarity.

## Response Generation

**4.3.1 Description and Priority**

Generate natural language responses based on the retrieved documents using transformer-based models (e.g., GPT-4). High priority as this feature provides the main functionality of the chatbot.

**4.3.2 Stimulus/Response Sequences**

**Stimulus**: The system retrieves relevant documents.

**Response**: The language model generates a human-like response, which is presented to the user.

**4.3.3 Functional Requirements**

REQ-1: Use **GPT-4** to generate coherent responses based on the retrieved documents.

REQ-2: Fine-tune the model if needed to improve response quality based on college-specific terminology and queries.

# Other Nonfunctional Requirements

**5.1 Performance Requirements**

Response time should be less than 2 seconds from query submission to response generation.

The system should handle at least 1000 concurrent user sessions without significant performance degradation.

**5.2 Safety Requirements**

The chatbot must avoid providing inappropriate or harmful responses.

Safeguards must be in place to handle inappropriate queries (e.g., profanity filters).

**5.3 Security Requirements**

User data, including queries, must be encrypted during transmission (using **HTTPS**).

User authentication is not required for basic query submissions, but authentication may be needed for accessing personalized information (e.g., exam results, course details).

**5.4 Software Quality Attributes**

**Usability**: The UI should be intuitive and easy to use for all users, including students, faculty, and visitors.

**Maintainability**: Code should follow clean coding practices and be modular for ease of updates and debugging.

**Reliability**: The chatbot should provide accurate responses and handle errors gracefully.

**5.5 Business Rules**

Only **students** and **faculty** can access sensitive or personal data, such as exam results or course grades.

The chatbot should always provide an appropriate answer, either by retrieving the relevant document or stating that the information is unavailable.

# Other Requirements

**Database**: The vector embeddings must be stored in a scalable database (FAISS).

**Language support**: Multilingual support may be added in future releases to support non-English speakers.