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

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

### Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

# Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

# Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

# Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>