**Software Requirements**

**Specification**

**for**

**Research Paper Summarizations using RAG Model**

**Version 2.0 approved**

**Prepared by**

**S. Satwika (23BD1A665R)**

**N. Gayatri (23BD1A665B)**

**B. Vaishnavi (23BD1A664A)**

**Keshav Memorial Institute of Technology**

**01-04-2025**

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

# Table of Contents

[Table of Contents ii](#_TOC_250033)

[Revision History ii](#_TOC_250032)

1. [Introduction 1](#_TOC_250031)
   1. [Purpose 1](#_TOC_250030)
   2. [Document Conventions 1](#_TOC_250029)
   3. [Intended Audience and Reading Suggestions 1](#_TOC_250028)
   4. [Product Scope 1](#_TOC_250027)
   5. [References 1](#_TOC_250026)
2. [Overall Description 2](#_TOC_250025)
   1. [Product Perspective 2](#_TOC_250024)
   2. [Product Functions 2](#_TOC_250023)
   3. [User Classes and Characteristics 2](#_TOC_250022)
   4. [Operating Environment 2](#_TOC_250021)
   5. [Design and Implementation Constraints 2](#_TOC_250020)
   6. [User Documentation 2](#_TOC_250019)
   7. [Assumptions and Dependencies 3](#_TOC_250018)
3. [External Interface Requirements 3](#_TOC_250017)
   1. [User Interfaces 3](#_TOC_250016)
   2. [Hardware Interfaces 3](#_TOC_250015)
   3. [Software Interfaces 3](#_TOC_250014)
   4. [Communications Interfaces 3](#_TOC_250013)
4. [System Features 4](#_TOC_250012)
   1. [System Feature 1 4](#_TOC_250011)
   2. [System Feature 2 (and so on) 4](#_TOC_250010)
5. [Other Nonfunctional Requirements 4](#_TOC_250009)
   1. [Performance Requirements 4](#_TOC_250008)
   2. [Safety Requirements 5](#_TOC_250007)
   3. [Security Requirements 5](#_TOC_250006)
   4. [Software Quality Attributes 5](#_TOC_250005)
   5. [Business Rules 5](#_TOC_250004)
6. [Other Requirements 5](#_TOC_250003)

[Appendix A: Glossary 5](#_TOC_250002)

[Appendix B: Analysis Models 5](#_TOC_250001)

[Appendix C: To Be Determined List 6](#_TOC_250000)

# Revision History

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

# Introduction

## Purpose

The purpose of this document is to define the software requirements for the **Research Paper Summarizations using RAG Model** project. This project aims to provide a web-based application that allows users to upload research papers in PDF/Docx format and receive a summarized version of the document, along with a Q&A feature for further exploration of the content. The system will use the RAG (Retrieval-Augmented Generation) model, FAISS, and LLM for summarization and query handling.

## Document Conventions

This document follows the IEEE SRS template. Requirements are numbered sequentially (e.g., REQ-1, REQ-2). Highlighted terms such as **RAG Model**, **LLM**, and **FAISS**  are key components of the system. Priorities for requirements are indicated as High, Medium, or Low.

## Intended Audience and Reading Suggestions

This document is intended for:

* **Developers**: To understand the system's functional and nonfunctional requirements.
* **Project Managers**: To oversee the project's scope and deliverables.
* **Testers**: To create test cases based on the requirements.
* **End Users**: To understand the system's capabilities and limitations.

## Product Scope

The product is a web-based application that allows users to upload research papers and receive a summarized version of the document. The system will use the RAG model to train on the uploaded data and provide a summary of approximately 300–600 words, including at least two advantages and disadvantages. Additionally, the system will support a Q&A feature for further exploration of the document's content. The application will be built using ReactJS for the front end, Flask for the back end, and MongoDB for database storage

## References

* **Langchain Framework Documentation**: <https://docs.langchain.com/>
* **Llama Documentation**: <https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>
* **FAISS Documentation**: <https://faiss.ai/>
* **ReactJS Documentation**: <https://reactjs.org/docs/getting-started.html>
* **Flask Documentation**: <https://flask.palletsprojects.com/>
* **MongoDB Documentation**: <https://www.mongodb.com/docs/>

# Overall Description

## Product Perspective

The product is a standalone web application that integrates with a RAG model for summarization and Q&A functionality. It is designed to assist researchers, students, and professionals in quickly understanding the key points of research papers. The system will be hosted on a cloud platform, and users will interact with it through a web browser.

## Product Functions

* **Research Paper Summarization**: Users can upload research papers in PDF/Docx format, and the system will generate a summary of approximately 300–600 words, including at least two advantages and disadvantages.
* **Query & Answer (Q&A)**: Users can ask questions related to the uploaded document, and the system will provide answers based on the trained RAG model.
* **PDF preview**: Users can view a preview of the uploaded PDF document within the system before generating a summary or querying the content.

## User Classes and Characteristics

* **Researchers**: Frequent users who need quick summaries of research papers.
* **Students**: Users who require summaries for academic purposes.
* **Professionals**: Users who need to quickly understand technical documents.

## Operating Environment

* **Front End**: ReactJS with Bootstrap.
* **Back End**: Python with Flask.
* **Database**: Cloud MongoDB.
* **Model**: RAG, Llama 3.3 70B Instruct Turbo Free.
* **Vector Storage**: FAISS.
* **Framework**: Langchain.

## Design and Implementation Constraint

* The system must support conversational memory for Q&A sessions.
* The application must be hosted on a cloud platform.

## User Documentation

* **User Manual**: Instructions for uploading documents, generating summaries, and using the Q&A feature.
* **Online Help**: FAQs within the application.

## Assumptions and Dependencies

* The RAG model will be pre-trained for summarization tasks.
* The system assumes that users will upload valid research papers in PDF or Docx format.
* The application depends on cloud hosting for MongoDB and the RAG model.

# External Interface Requirements

## User Interfaces

* **Landing Page**: A homepage with an overview of the application, including a brief description, key features (summarization and Q&A), and a call-to-action to log in or sign up.
* **Login Page**: A secure form for users to enter credentials (username/email and password) to access the application, with options for registration and password recovery.
* **Chat Interface**: A section for uploading research papers in PDF/Docx format, to display the generated summary, including advantages and disadvantages. And a text box for users to input questions and a section to display answers.

## Hardware Interfaces

* **N/A**: The system is web-based and does not require specific hardware interfaces.

## Software Interfaces

* **Front End**: ReactJS with Bootstrap.
* **Back End**: Python with Flask.
* **Database**: Cloud MongoDB.
* **Model**: RAG, Llama 3.3 70B Instruct Turbo Free.
* **Vector Storage:** FAISS.

## Communications Interfaces

* **HTTP/HTTPS**: For communication between the front end and back end.
* **API Calls**: For interactions between the back end and the RAG model.

# System Features

## Research Paper Summarization

### ***Description and Priority***

### ***Description:*** *The system will generate a summary of approximately 300- 600 words, including at least two advantages and disadvantages.*

### *Priority: High*

### ***Stimulus/Response Sequences***

### *User uploads a research paper.*

### *System processes the document and generates a summary.*

### *System displays the summary to the user.*

### ***Functional Requirements***

* **REQ-1**: The system must accept PDF and Docx file formats.
* **REQ-2**: The system must generate a summary of approximately 300- 600 words.

## Query & Answer (Q&A) Functionality

### 4.*2.1* ***Description and Priority***

### ***Description:*** *Users can ask questions related to the uploaded document, and the system will provide answers based on the trained RAG model.*

### *Priority: High*

### ***Stimulus/Response Sequences***

### *User inputs a question.*

### *System processes the question and generates an answer.*

### *3. System displays the answer to the user.*

### *4.2.3* ***Functional Requirements***

### *REQ-3: The system must support conversational memory for Q&A sessions.*

### *REQ-4: The system must provide accurate answers based on the uploaded document.*

# Other Nonfunctional Requirements

## Performance Requirements

* The system must generate summaries within 15 seconds of document upload.
* The Q&A feature must provide answers within 10 seconds of user input.

## Safety Requirements

* **N/A**: The system does not involve physical safety concerns.

## Security Requirements

* User data (uploaded documents) must be stored securely in MongoDB.
* API keys and sensitive information must be encrypted.

## Software Quality Attributes

* **Usability**: The system must be easy to use, with clear instructions for uploading documents and asking questions.
* **Reliability**: The system must generate accurate summaries and answers.
* **Maintainability**: The code must be well-documented for future updates.

## Business Rules

## *N/A: The system does not enforce specific business rules.*

# Other Requirements

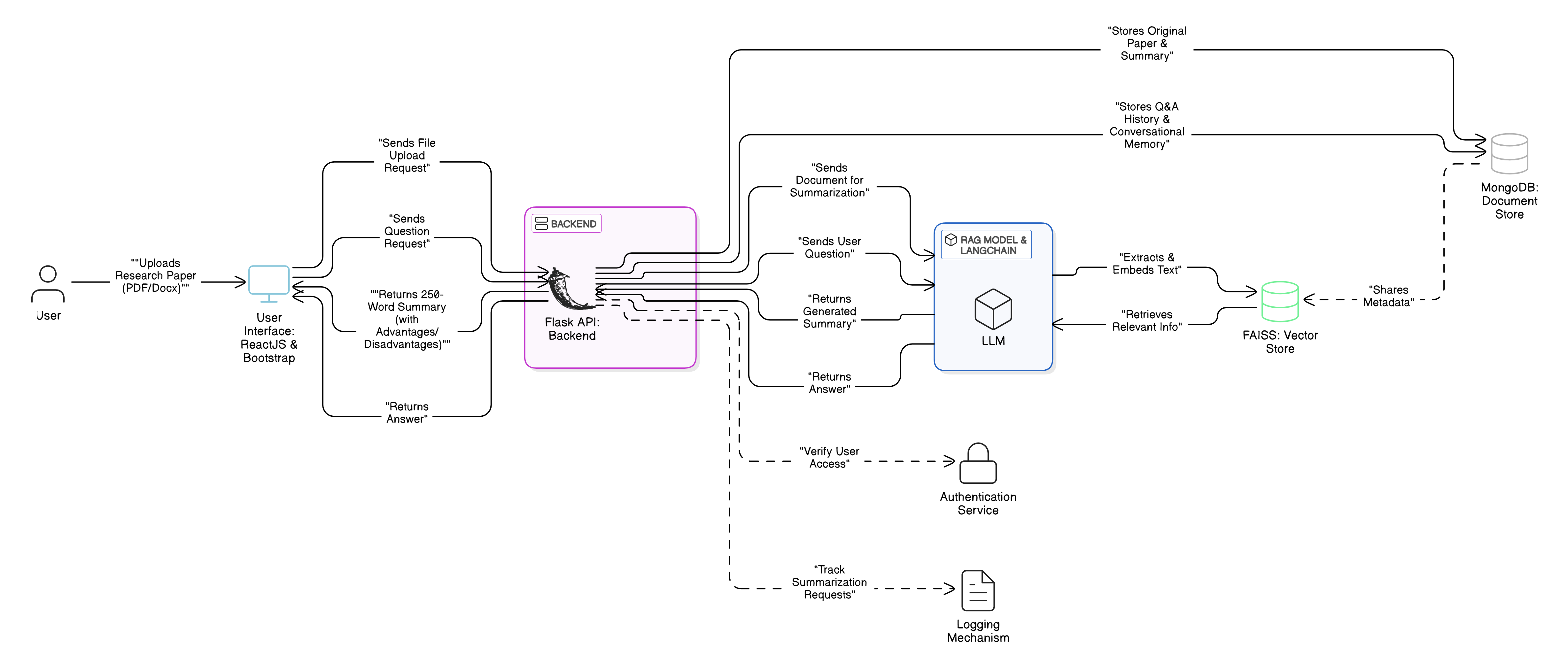
# *N/A: No additional requirements at this time.*

# Appendix A: Glossary

* **RAG Model**: Retrieval-Augmented Generation model used for summarization and Q&A.
* **Llama 3.3 70B Instruct Turbo** **Free**: A transformer-based model used for text generation.
* **FAISS**: A vector database used for storing and retrieving document embeddings.
* **Langchain**: A framework for building applications with language models.

# Appendix B: Analysis Models

WORK FLOW DIAGRAM:



ARCHITECTURE DIAGRAM:

