**A Real-time Research Project / Societal Related Project**

**Report on**

### Research Paper Summarization using RAG Model

Submitted in Partial fulfillment of requirements for B.Tech II Year II Semester course

**By**

**S. Satwika 23BD1A665R**

**N. Gayatri 23BD1A665B**

**B. Vaishnavi 23BD1A664A**

**Under the guidance of**

***Mr.Shanker***

***Assistant Professor, CSE***

***& Ms. J. Kamal Vijetha***

***Assistant Professor, CSE***



|  |  |
| --- | --- |
| |  | | --- | | **KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY  (AN AUTONOMOUS INSTITUTION)  Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH.** Narayanaguda, Hyderabad, Telangana-29 **2024-25** | |

**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY   
(AN AUTONOMOUS INSTITUTION)  
Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH**

### Narayanaguda, Hyderabad, Telangana-29

**CERTIFICATE**

This is to certify that this is a bonafide record of the project report titled **“Research Paper Summarization using RAG model”** which is being presented as the **Real-time Research Project / Societal Related Project** report by

1. **S. Satwika 23BD1A665R**
2. **N. Gayatri 23BD1A665B**
3. **B. Vaishnavi 23BD1A664A**

In partial fulfillment for the II Year II Semester Course RTRP in KMIT affiliated to the Jawaharlal Nehru Technological University Hyderabad, Hyderabad

**Mentors**

**Mr. Shanker Program Coordinator**

**Ms. J. Kamal Vijetha (Mr Shailesh Gangakhedkar)**

Submitted for Final Project Review held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Vision & Mission of KMIT**

**Vision of KMIT**

* + To be the fountainhead in producing highly skilled, globally competent engineers.
  + Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

# Mission of KMIT

* + To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms and prepares students to become successful professionals.
  + To establish an industry institute Interaction to make students ready for the industry.
  + To provide exposure to students on the latest hardware and software tools.
  + To promote research-based projects/activities in the emerging areas of technology convergence.
  + To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
  + To induce a spirit of nationalism which will enable the student to develop, understand India's challenges and to encourage them to develop effective solutions.
  + To support the faculty to accelerate their learning curve to deliver excellent service to students.

# PROGRAM OUTCOMES (POs)

**PO1. Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2. Problem Analysis:** Identify formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

**PO3. Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4. Conduct Investigations of Complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5. Modern Tool Usage:** Create select, and, apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6. The Engineer and Society:** Apply reasoning informed by contextual knowledge to societal, health, safety. Legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.

**PO7. Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

**PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO9. Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

**PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11. Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12. Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROJECT OUTCOMES**

**P1:** Develop a web-based application that enables users to upload research papers in PDF or DOCX format and obtain AI-generated summaries and query responses using a Retrieval-Augmented Generation (RAG) model.

**P2:** Implement secure user authentication and chat history persistence to enhance user experience and data security.

**P3:** Integrate advanced AI technologies, including LangChain and the Together API, to process documents and generate accurate, contextually relevant responses.

**P4:** Provide a user-friendly interface with features like document preview, chat manage ment, and real-time query processing to support academic and professional research

**MAPPING PROJECT OUTCOMES WITH PROGRAM OUTCOMES**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PO** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| **P1** | H | M | H | M | H | L | M | M | H | M | M | H |
| **P2** | M | L | M | L | M | M | L | H | H | M | H | M |
| **P3** | H | H | H | H | H | M | M | M | M | H | M | H |
| **P4** | M | M | H | M | H | H | M | H | H | H | H | M |

L – LOW M –MEDIUM H– HIGH

# DECLARATION

We hereby declare that the results embodied in the dissertation entitled **“**Research Paper Summarization using RAG Model” has been carried out by us together during the academic year 2024-25 as a partial fulfillment of the B.Tech II Year II Semester Course “**Real-time Research Project / Societal Related Project**”. We have not submitted this report to any other Course/College.



### Student Name Rollno.

1. **S. Satwika 23BD1A665R**
2. **N. Gayatri 23BD1A665B**
3. **B. Vaishnavi 23BD1A664A**

# ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to **Dr. B L Malleswari**, Principal who encouraged us to do the Project.

We are grateful to **Mr. Neil Gogte,** Founder & Director and **Mr. S. Nitin,** Director,for facilitating all the amenities required for carrying out this project.



We express our sincere gratitude to **Ms. Deepa Ganu**, Director Academic for providing an excellent environment in the college.

We are also thankful to **Mr.Shailesh Gangakhedkar**, **Real-Time Research Project** Program Coordinator for providing us with time to make this project a success within the given schedule.

We are also thankful to our Project Mentor **Ms. J. Kamal Vijetha & Mr. Shanker**, for her/his valuable guidance and encouragement given to us throughout the project work.

We would like to thank the entire KMIT faculty, who helped us directly and indirectly in the completion of the project.

We sincerely thank our friends and family for their constant motivation during the project work.

## Student Name Roll no.

**S. Satwika 23BD1A665R**

**N. Gayatri 23BD1A665B**

**B. Vaishnavi 23BD1A664A**

# ABSTRACT

The Research Paper Summarization project develops a web-based application that lever ages a Retrieval-Augmented Generation (RAG) model integrated with Large Language Models (LLMs) to summarize and query research papers in PDF and DOCX formats. The system allows users to upload documents, receive AI-generated summaries, and engage in follow-up question-and-answer sessions, streamlining academic and professional research. Built with a ReactJS frontend, Flask backend, and MongoDB database, it incorporates secure user authentication, chat history persistence, and document preview features. The RAG model, powered by Llama 3.3 70B Instruct Turbo Free via the Together API, uses FAISS for vector storage and LangChain for AI orchestration, ensuring accurate and contextually relevant responses. The application addresses limitations of traditional summarization methods, such as factual inaccuracies and lack of coherence, by combining retrieval and generation techniques. Challenges like handling diverse document formats and managing API rate limits were overcome through robust error handling and optimized text extraction using tools like pdfplumber and python-docx. This project aligns with KMIT’s mission to promote research-based activities and provides a scalable, user-friendly tool for researchers, students, and professionals.

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name of Screenshot** | **Page No.** |
| **1.** |  |  |
| **2.** |  |  |
| **3.** |  |  |
| **4.** |  |  |
| **5.** |  |  |

**CONTENTS**

DESCRIPTION PAGE [CHAPTER - 1 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2et92p0)

* 1. [INTRODUCTION 2-3](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.tyjcwt)
     1. Purpose of the project 2
     2. Problem with Existing Systems 2
     3. [Proposed System 2](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3dy6vkm)
     4. [Scope of the Project](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.4d34og8) 2
     5. [Architecture Diagram 3](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2s8eyo1)

[CHAPTER – 2 4](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.17dp8vu)

* 1. [LITERATURE SURVEY 5](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3rdcrjn)

[CHAPTER - 3 6](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.26in1rg)

* 1. SOFTWARE REQUIREMENT SPECIFICATION 7-8
     1. [Introduction to SRS 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.lnxbz9)
     2. [Role of SRS 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.35nkun2)
     3. [Requirements Specification Document 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.1ksv4uv)
     4. [Functional Requirements 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.44sinio)
     5. [Non-Functional Requirements 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2jxsxqh)
     6. [Performance Requirements 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.z337ya)
     7. [Software Requirements 7](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3j2qqm3)
     8. [Hardware Requirements](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.1y810tw) 8

[CHAPTER – 4](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.4i7ojhp) 9

* 1. [SYSTEM DESIGN 10-](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2xcytpi)14
     1. [Introduction to UML 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.1ci93xb)0
     2. [UML Diagrams 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3whwml4)0
        1. [Use Case Diagram 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2bn6wsx)0
        2. [Sequence Diagram 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.qsh70q)1
        3. State Chart [Diagram 1](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3as4poj)2
        4. [Deployment Diagram](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2p2csry) 13
     3. [TECHNOLOGIES USED](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.147n2zr) 14

[CHAPTER – 5](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3o7alnk) 15

* 1. IMPLEMENTATION 16-20
     1. Setting up connections with Raspberry PI 16
     2. Coding the logic 16
     3. Connecting the dashboard 16
     4. UI Screenshots 16

[CHAPTER – 6](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.23ckvvd) 21

* 1. SOFTWARE TESTING 22-23
     1. [Introduction](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.ihv636) 22
        1. [Testing Objectives](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.32hioqz) 22
        2. [Testing Strategies](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.1hmsyys) 22
        3. [System Evaluation](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.41mghml) 22
        4. Testing New System 22
     2. [Test Cases](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2grqrue) 22

[CONCLUSION](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.4f1mdlm) 24

[FUTURE ENHANCEMENTS](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.2u6wntf) 25

[REFERENCES](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.19c6y18) 26

[BIBLIOGRAPHY](https://docs.google.com/document/d/1vNwnyGx9h1yMU6_gVJ2-YRqX4bTSdHyv/edit#heading%3Dh.3tbugp1) 27

# CHAPTER-1

# INTRODUCTION

# Purpose of the Project

# The purpose of this project is to develop a web-based application that automates the summarization and querying of research papers using a Retrieval-Augmented Generation (RAG) model integrated with Large Language Models (LLMs). The system, named KMITPaper, aims to assist researchers, students, and professionals by providing concise, accurate summaries and interactive question-and-answer capabilities for uploaded PDF and DOCX documents. By leveraging advanced AI technologies, the project seeks to reduce the time and effort required for manual literature review, enhance comprehension of complex academic texts, and support decision-making in academic and professional settings. The application aligns with KMIT’s vision of producing globally competent engineers and promoting research-based projects.

# Problem with Existing Systems

# Traditional research paper summarization methods, such as manual reading or basic NLP models, face several limitations. Manual summarization is time-consuming and impractical for large volumes of literature. Early NLP models, like extractive summarizers, often fail to capture contextual nuances, producing incoherent or factually inaccurate sum maries. Advanced models like BART or Pegasus, while improved, struggle with long documents and may generate “hallucinations”—information not present in the source text. Additionally, existing tools lack interactive querying capabilities, limiting their utility for users seeking specific insights. These systems often do not support diverse document formats or provide user-friendly interfaces, making them less accessible to non-technical users.

# Proposed System

# The proposed system, KMITPaper, addresses these challenges by integrating a RAG model with LLMs, specifically Llama 3.3 70B Instruct Turbo Free, to generate high quality summaries and responses. The system allows users to upload research papers in PDF or DOCX format (up to 10MB), extracts text and metadata using tools like pdfplumber and python-docx, and processes documents into 1500-character chunks with 200-character overlap for efficient retrieval. FAISS and HuggingFace embeddings enable accurate document retrieval, while the Together API facilitates LLM-based generation. The application features secure JWT-based authentication, chat session management, and document preview, all accessible via a ReactJS frontend and Flask backend with MongoDB storage. This system outperforms conventional methods by ensuring factual accuracy, coherence, and interactivity.

# Scope of the Project

# The project focuses on developing a web-based tool for summarizing and querying research papers, targeting researchers, students, and professionals. Key features include user authentication, document upload and processing, AI-driven summarization, interactive Q&A, chat management, and file preview. The system supports PDF and DOCX files up to 10MB and assumes stable internet access and extractable text in documents. Future enhancements may include mobile app development, support for additional file formats, and improved performance for large files. The project does not cover physical hardware interfaces or non-text document processing (e.g., scanned images). It aligns with KMIT’s mission to provide exposure to modern tools and promote research-based activities.

# Architecture Diagram

# The system architecture comprises a ReactJS frontend for user interaction, a Flask back end for processing requests, and a MongoDB database for storing user data, chat sessions, and document metadata.

# The frontend communicates with the backend via HTTP re quests, while the backend interfaces with the Together API for LLM responses and FAISS for vector storage. The RAG model splits documents into chunks, generates embeddings, and retrieves relevant sections to inform LLM outputs. Authentication is handled us ing JWT tokens, ensuring secure access to protected routes. The architecture ensures scalability, modularity, and efficient data flow between components.

# CHAPTER-2

# Literature Survey

# The development of KMITPaper builds on extensive research in natural language processing (NLP) and document summarization. Early summarization techniques relied on extractive methods, selecting key sentences based on statistical features like TF-IDF. However, these approaches, as discussed in Nenkova and McKeown (2011), often produced disjointed outputs lacking coherence. Abstractive models like BART (Lewis et al., 2020) and Pegasus (Zhang et al., 2020) improved coherence by generating human-like summaries but struggled with factual consistency in long documents. The introduction of Retrieval-Augmented Generation (RAG) by Lewis et al. (2020) marked a significant advancement. RAG combines retrieval mechanisms with generative models, enabling contextually relevant responses by fetching document chunks before generation. This approach outperforms traditional models in knowledge-intensive tasks like question answering, as demonstrated in the original RAG paper, which achieved state-of-the-art results on three open-domain QA datasets. The use of FAISS for efficient vector storage, as explored in Johnson et al. (2021), further enhances retrieval speed and accuracy. Recent studies, such as those by Gao et al. (2023), highlight the effectiveness of LLMs like Llama in summarization tasks when fine-tuned with domain-specific data. However, challenges like API rate limits and handling complex document layouts (e.g., tables) persist. KMITPaper leverages these advancements, using LangChain for AI orchestration and pdfplumber for robust text extraction, addressing gaps in existing systems by providing an interactive, user-friendly tool tailored for academic research.

# 

# CHAPTER- 3

# 3. Software Requirement Specification

# 3.1 Introduction to SRS

# The Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for KMITPaper, a web-based application for research paper summarization using a RAG model. The SRS serves as a blueprint for developers, project managers, testers, and end users, ensuring alignment on system capabilities and constraints. This covers the full system, including frontend, backend, and database components.

# 3.2 Role of SRS

# The SRS defines the project’s scope, functionalities, and performance criteria, facilitating clear communication among stakeholders. It guides developers in implementing features like document upload and query processing, enables testers to design test cases, and informs users about system capabilities. By specifying requirements like JWT authentication and file size limits, the SRS ensures the system meets academic and professional needs while maintaining security and usability.

# 3.3 Requirements Specification Document

# The SRS is structured into sections covering introduction, overall description, external interface requirements, system features, and non-functional requirements. It uses conventions like bold headings, italicized terms, and high-priority defaults to ensure clarity. The document references external resources, such as MongoDB Atlas and Flask docu mentation, to support implementation details

# 3.4 Functional Requirements

**• User Authentication:** The system must support user registration and login with JWT tokens valid for 1 hour, hashing passwords using bcrypt, and rejecting duplicate email registrations (REQ-1, REQ-2, REQ-3).

• **Document Upload and Processing:** The system must accept PDF and DOCX files up to 10MB, extract text and metadata using pdfplumber or python-docx, and store files with unique filenames (REQ-4, REQ-5, REQ-6).

• **Chat Management:** The system must persist chat sessions in MongoDB, allow renaming with non-empty names, and support pinning for easy access (REQ-7, REQ-8, REQ-9).

• **Query Processing:** The system must split documents into 1500-character chunks with 200-character overlap, use FAISS and HuggingFace embeddings for retrieval, and integrate with the Together API for LLM responses (REQ-10, REQ-11, REQ 12).

**3.5 Non-Functional Requirements**

• **Usability**: The system must provide clear instructions and a user-friendly interface for uploading documents and querying.

• **Reliability**: The system must generate accurate summaries and responses based on uploaded documents.

• **Maintainability**: The code must be well-documented to facilitate future updates.

**3.6 Performance Requirements**

The system must process a 10MB document and return a response, ensuring efficient summarization and querying for users.

**3.7 Software Requirements**

• **Frontend**: ReactJS, Bootstrap, Custom CSS.

• **Backend**: Python 3.8+, Flask, LangChain, pdfplumber, python-docx, FAISS, HuggingFace transformers.

• **Database**: MongoDB Atlas.

• **API**: Together API for Llama 3.3 70B Instruct Turbo Free.

• **OS**: Cross-platform (Windows, Linux, macOS for development).

**3.8 Hardware Requirements**

• **Development**: Minimum 8GB RAM, 2GHz processor, 20GB storage.

• **Client:** Modern web browser (Chrome, Firefox) on any device with 4GB RAM.

# CHAPTER-4

# System Design

# 4.1 Introduction to UML

Unified Modeling Language (UML) provides a standardized way to visualize and document the system’s architecture and behavior. For KMITPaper, UML diagrams illustrate user interactions, data flow, system states, and deployment, aiding developers in implementing the system and stakeholders in understanding its design.

**4.2 UML Diagrams**

**4.2.1 Use Case Diagram**

The use case diagram outlines interactions between actors (users) and the system. Actors include registered users and guest users. Use cases include:

• **Register**: Users create an account with email, username, and password.

• **Login**: Users authenticate using credentials to access features.

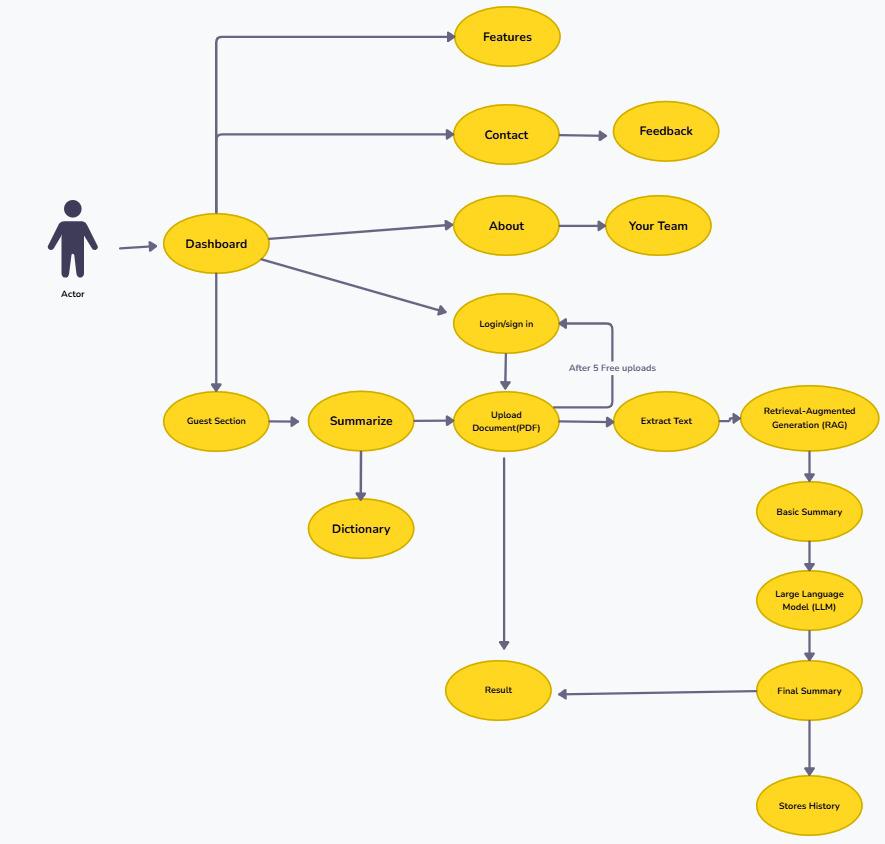
• **Upload Document**: Users upload PDF/DOCX files for processing.

• **Query** **Document**: Users submit questions to receive AI-generated responses.

• **Manage Chats**: Users create, delete, rename, or pin chat sessions.

• **Preview Document**: Users view uploaded file excerpts.

Registered users have access to all features, while guest users are limited to ephemeral chats without persistence.

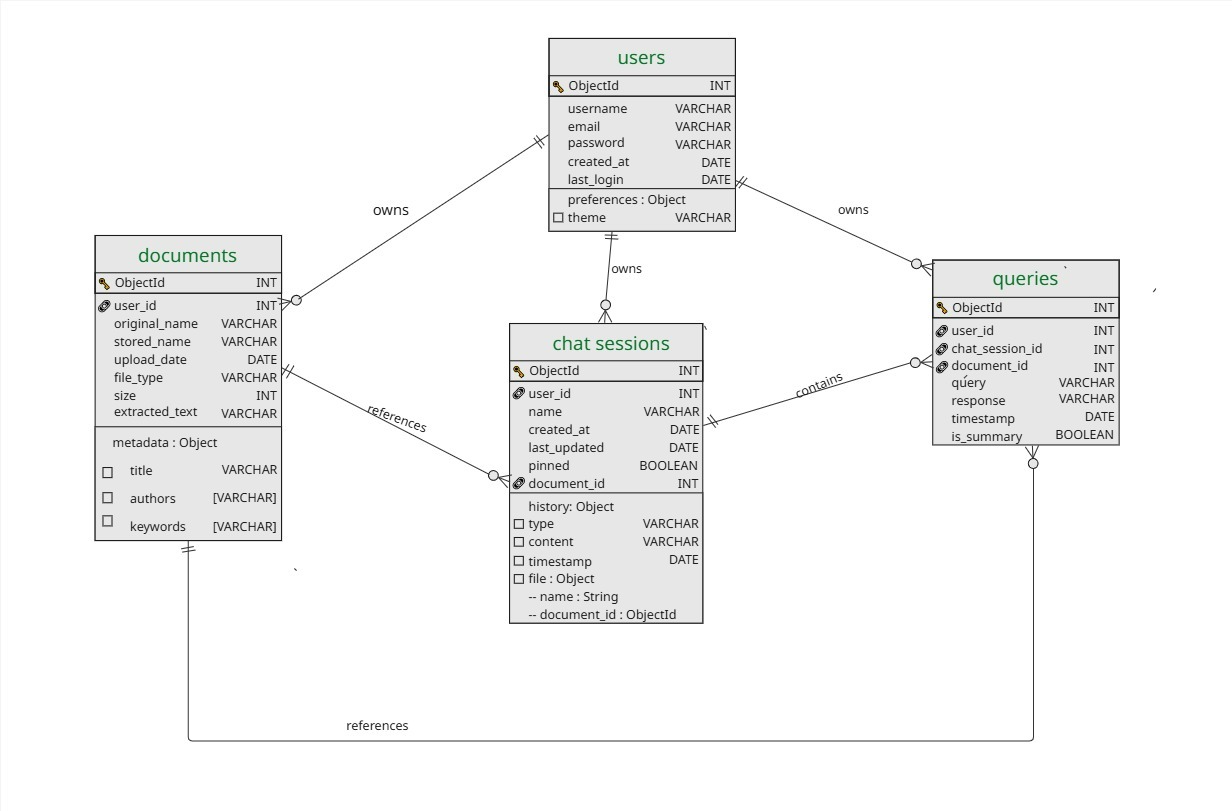


### ****4.2.2 Database Diagram****

The database schema includes four main collections: users, documents, chat\_sessions, and queries.

* + **Users:** store credentials and preferences.
  + **Documents:** include metadata like file name, type, size, and extracted text.
  + **Chat Sessions:** track chat histories linked to users and documents.
  + **Queries:** store user questions and AI responses.

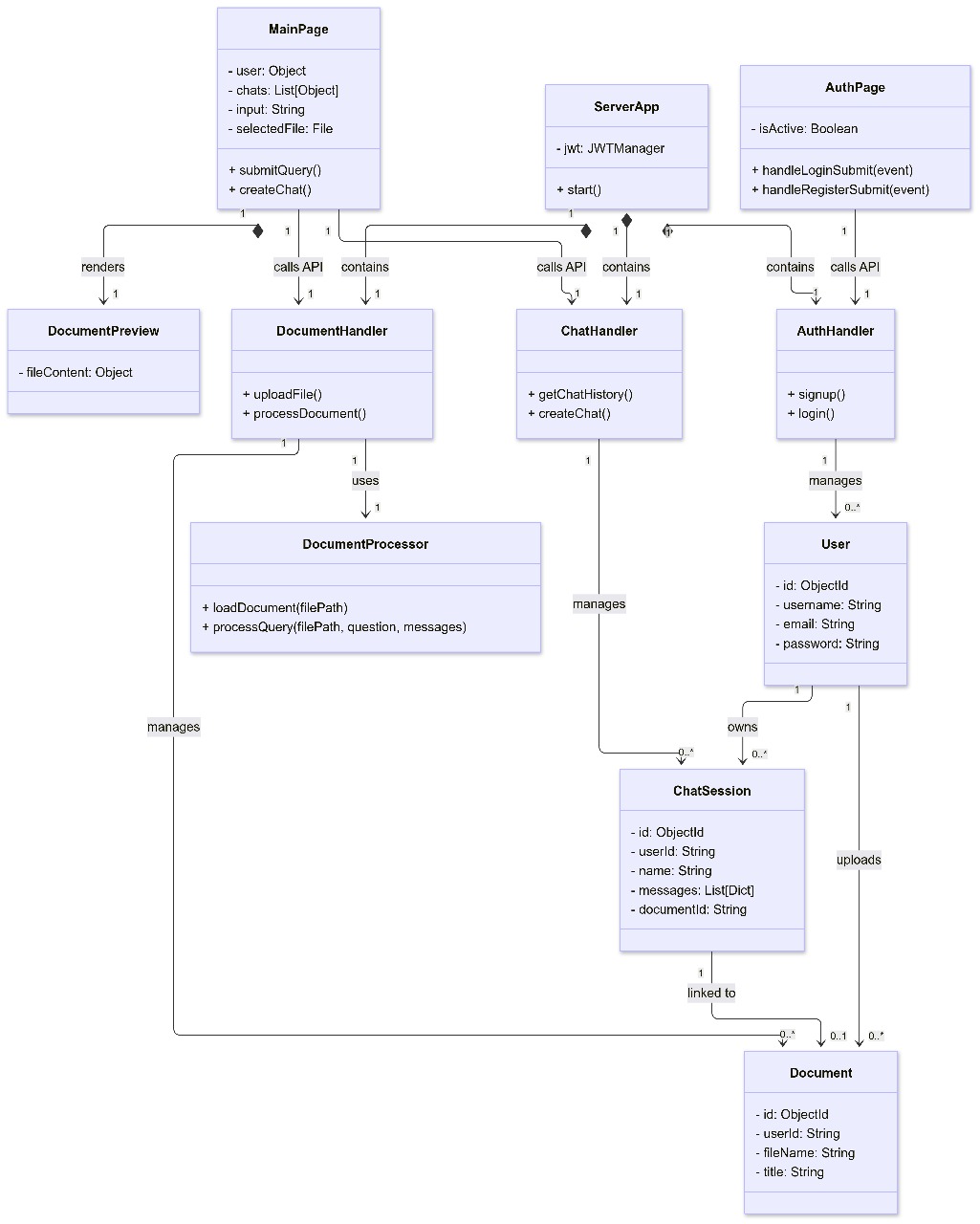
Relations: Users own documents and chat sessions. Chat sessions reference documents and contain queries. Queries are linked to both users and documents.



**4.2.3 Class Diagram**

The class diagram outlines frontend-backend interaction logic.

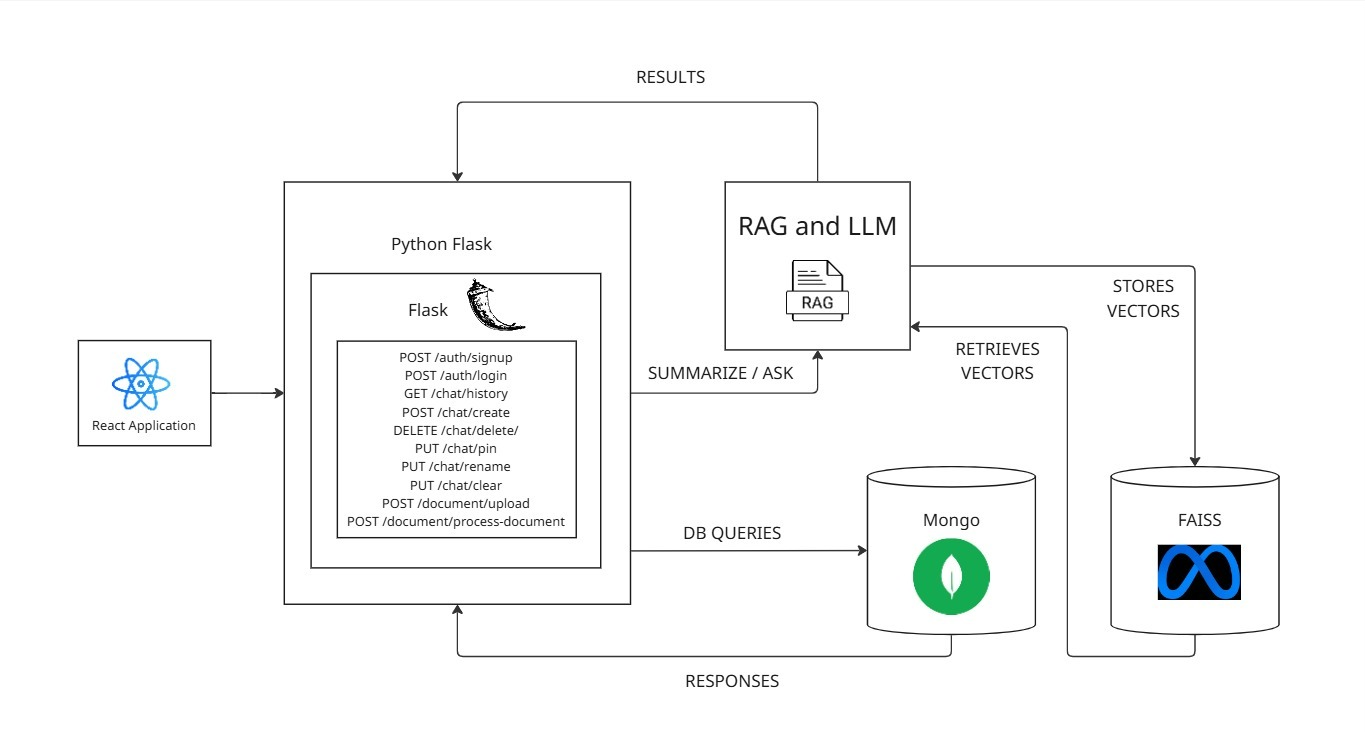
* + **MainPage** handles user inputs, chat creation, and query submission.
  + **Handlers** like AuthHandler, ChatHandler, and DocumentHandler connect to the API.
  + **DocumentProcessor** manages loading documents and query processing.
  + **AuthPage** manages login/register views.
  + Classes like User, ChatSession, and Document model the stored data.  
    Composition and aggregation indicate object relationships and API dependency.



**4.2.4 Architecture Diagram**

The architecture integrates a **React frontend** with a **Python Flask backend**.

* + Flask handles API routes for authentication, uploading, summarizing, and asking questions.
  + **RAG and LLM** components process document summarization and Q&A.
  + **MongoDB** stores document data; **FAISS** stores and retrieves vector embeddings.  
    Data flows from React to Flask, then to LLM/FAISS, and results return to the user.



**4.3 Technologies** **Used**

• **Frontend**: ReactJS for dynamic UI, Bootstrap & Custom CSS for styling

• **Backend**: Flask for RESTful API, Python for processing logic, LangChain for AI orchestration.

• **Database**: MongoDB for storage.

• **AI:** Llama 3.3 70B Instruct Turbo Free via Together API, FAISS for vector storage, HuggingFace embeddings for text representation.

• **Text Extraction**: pdfplumber for PDFs, python-docx for DOCX files.

• **Authentication**: JWT for secure access, bcrypt for password hashing.

# CHAPTER-5

# Implementation

# 5.1 Setting up Connections with MongoDB and Together API

The implementation begins with configuring connections to external services. MongoDB Atlas is set up using db.py, which initializes a client with a connection string stored in environment variables. The database stores collections for users, chats, and documents, with indexes on email and chat IDs for efficient queries. The Together API is configured in server.py, using an API key to access Llama 3.3 70B Instruct Turbo Free. The backend uses LangChain to orchestrate RAG workflows, ensuring seamless integration between retrieval and generation.

**5.2 Coding the Logic**

The backend logic is distributed across multiple modules:

• **auth.py**: Handles user registration (register\_user) and login (login\_user), hashing passwords with

bcrypt and issuing JWT tokens valid for 1 hour. Duplicate email checks prevent registration errors.

• **file\_utils.py**: Manages file uploads (upload\_file) by validating format (PDF/DOCX) and size (<10MB), saving files with unique UUID-based names to prevent overwrites.

• **nlp\_utils.py**: Processes documents (process\_document) by extracting text and metadata, splitting into 1500-character chunks with 200-character overlap, and gen erating embeddings using HuggingFace’s sentence-transformers.

• **document.py**: Handles document queries (process\_document\_query) by retrieving relevant chunks via FAISS and generating responses using the Together API.

• **chat.py**: Manages chat sessions (create\_chat, delete\_chat, pin\_chat), storing in teractions in MongoDB and supporting renaming and pinning.

• **server.py**: Defines Flask routes (e.g., /api/register, /api/upload, /api/query) and integrates all modules, ensuring secure access with JWT validation

The frontend, implemented in Mainpage.jsx, uses React components to render the landing page, login form, chat interface, and document preview. State management handles user sessions, chat lists, and query responses, with Axios facilitating API calls.

**5.3 Connecting the Dashboard**

The React frontend serves as the dashboard, providing a chat-style interface for users to upload documents, view summaries, and submit queries. The Mainpage.jsx componentorchestrates UI elements, including a file upload form, chat session list, and query input box. The dashboard integrates with the backend via RESTful API endpoints, ensuring seamless data flow between user actions and server responses.

**5.4 UI Screenshots**

The system includes key UI components:

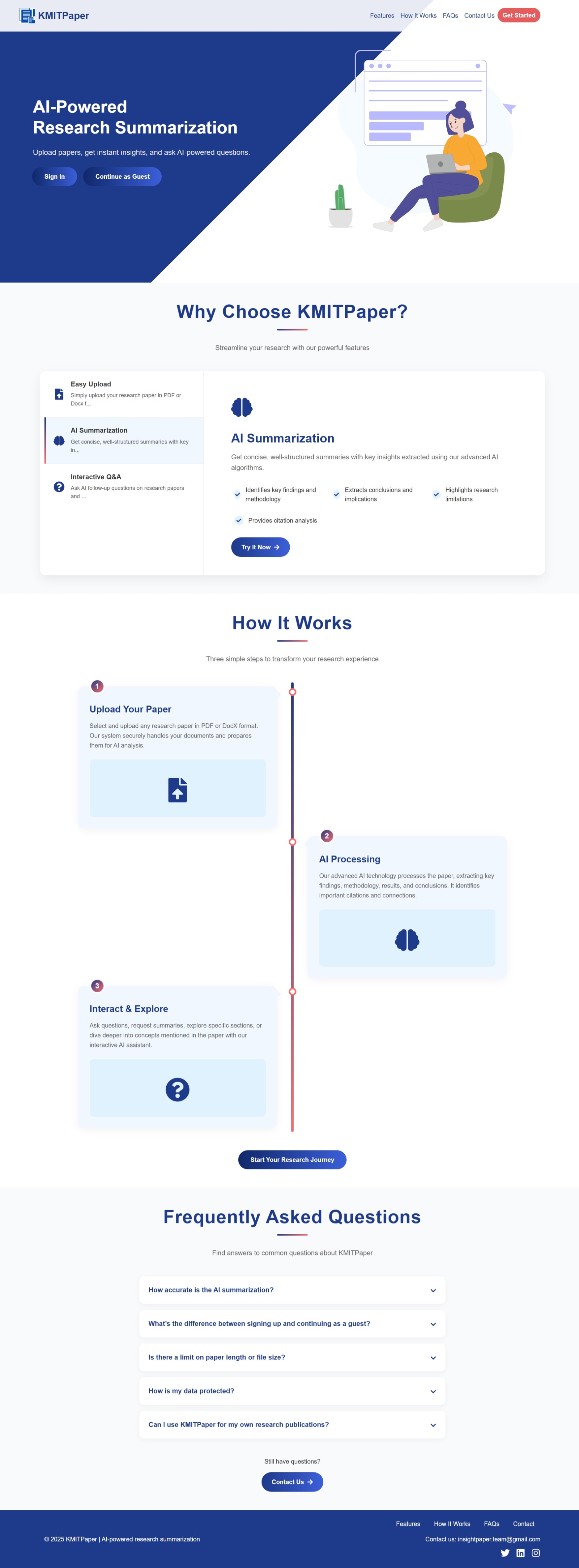
• **Landing Page**: Displays project overview and login/signup options.

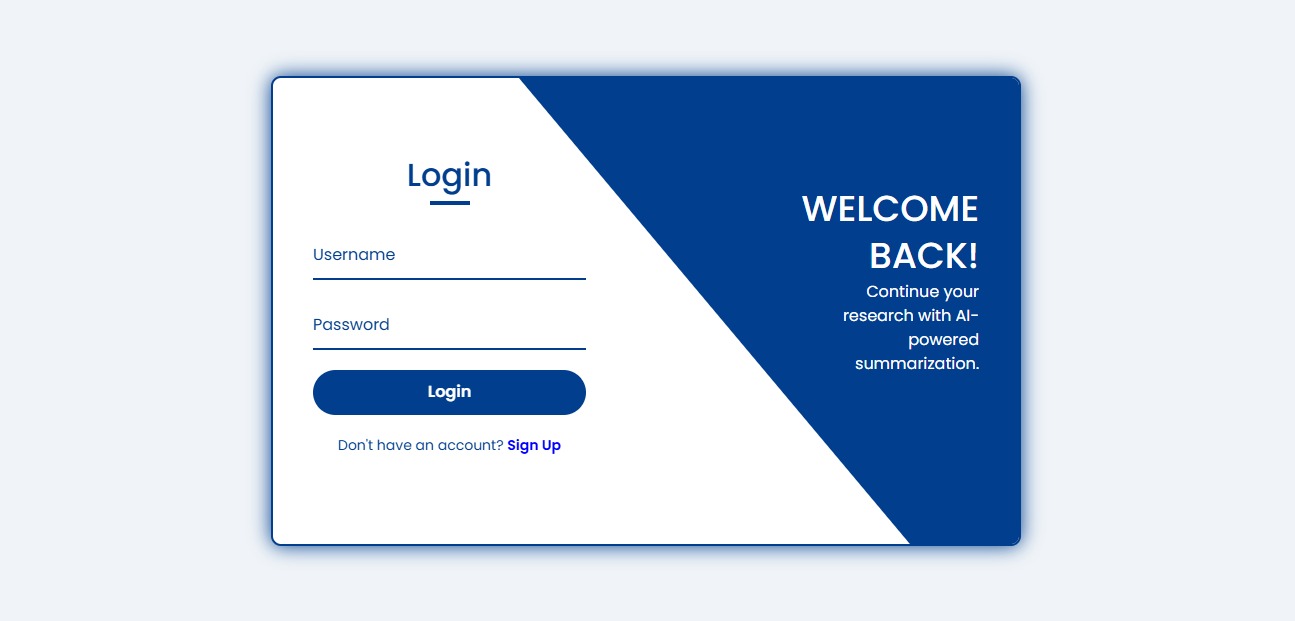
• **Login Page**: Secure form for user authentication.

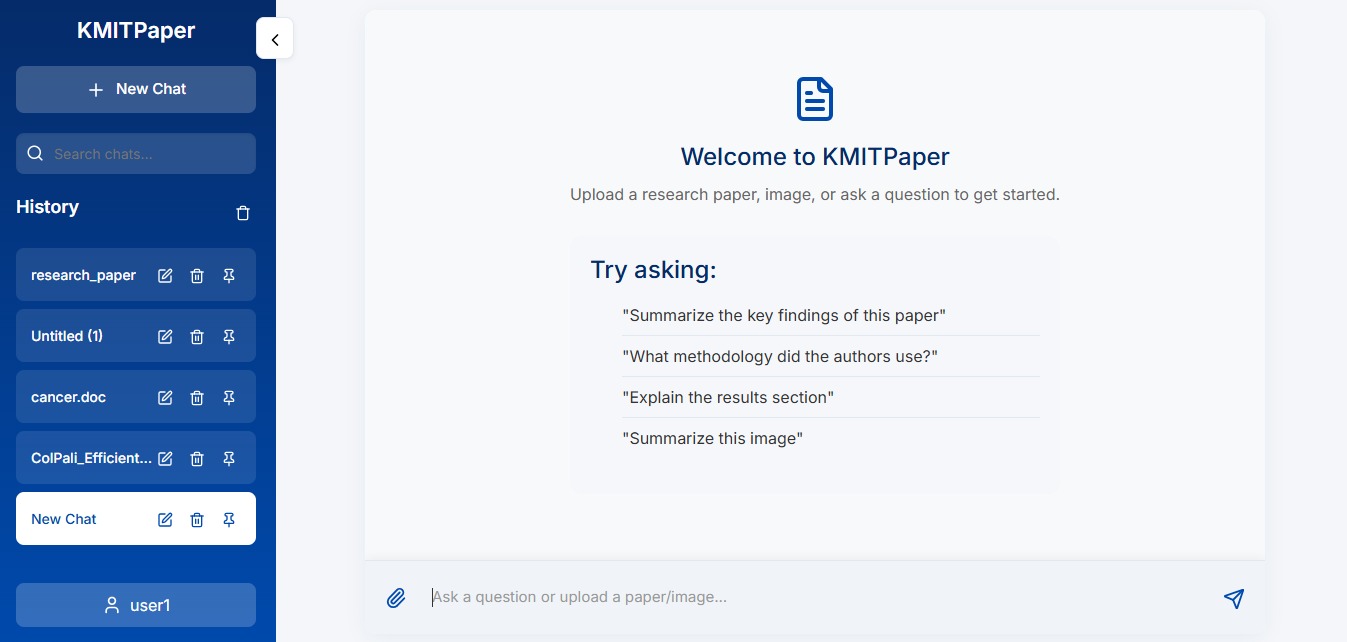
• **Chat Interface**: Allows document uploads, query input, and response display.

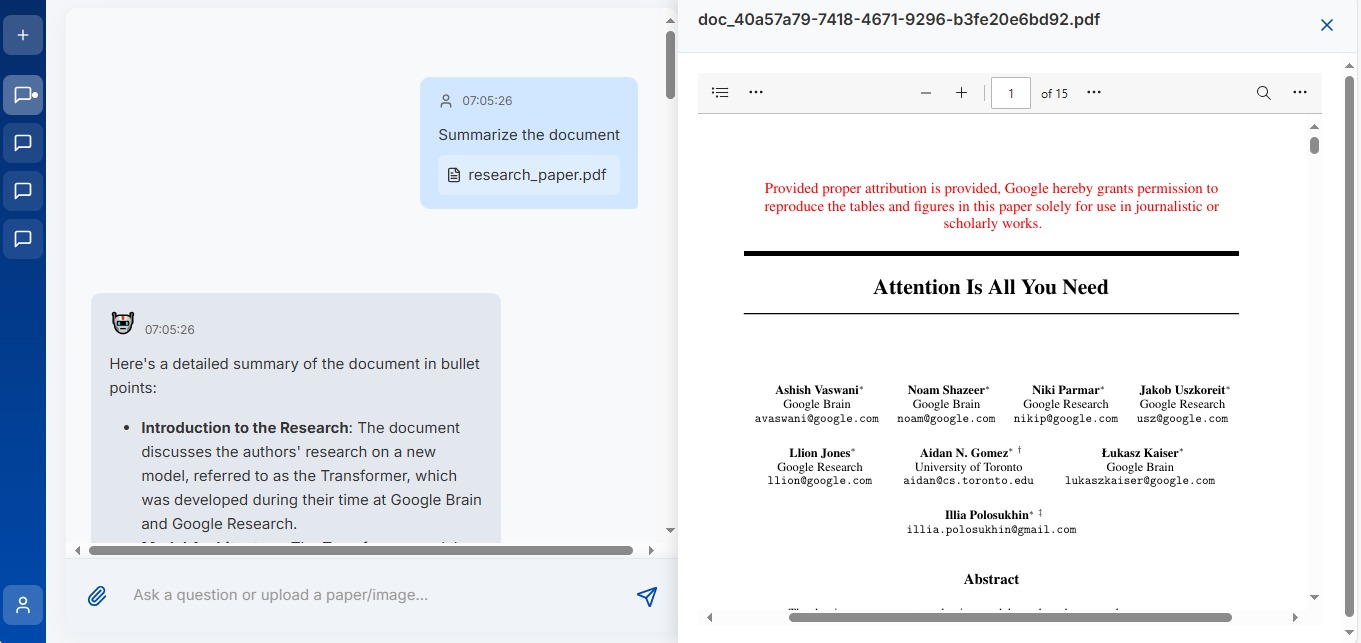
• **Document Preview**: Shows excerpts of uploaded files.

The UI is designed for usability, with Bootstrap ensuring responsiveness. The chat interface mimics modern messaging apps, with pinned chats highlighted and a clear distinction between user queries and AI responses. Tooltips guide users through document upload and query processes, enhancing accessibility.









# CHAPTER-6

# Software Testing

**6.1 Introduction**

Software testing ensures KMITPaper meets its functional and non-functional requirements, delivering a reliable and user-friendly experience.

**6.1.1** **Testing Objectives**

• Verify that all functional requirements (e.g., document upload, query processing) work as specified.

• Ensure non-functional requirements like performance and security are met.

• Identify and fix bugs to enhance system reliability

**6.1.2 Testing Strategies**

• **Unit Testing**: Tests individual functions (e.g., process\_document in nlp\_utils.py) using Pytest, covering edge cases like invalid file formats.

• **Integration Testing**: Verifies interactions between modules, such as file upload and query processing, ensuring data flows correctly between frontend, backend, and database.

• **System Testing**: Tests the entire application, including user authentication, chat management, and AI response accuracy.

• **Performance Testing**: Measures response times for 10MB document processing.

**6.1.3 System Evaluation**

The system was evaluated using sample research papers from arXiv, assessing summary accuracy and query response relevance. User feedback highlighted the need for better error messages.

**6.1.4 Testing New System**

Testing involved simulating real-world scenarios, such as uploading malformed PDFs or complex queries, to ensure robustness. The system handled most cases effectively.

**6.2 Test Cases**

• TC1:

User Registration

– Input: Valid email, username, password.

– Expected Output: User created, 200 OK.

– Result: Pass.

• TC2:

Document Upload

– Input: 5MB PDF file.

– Expected Output: File stored, text extracted, 200 OK.

– Result: Pass.

• TC3:

Query Processing

– Input: Query “What is the main contribution?” on uploaded paper.

– Expected Output: Relevant summary/response within 30 seconds.

– Result: Pass.

• TC4:

Chat Deletion

– Input: Delete chat session.

– Expected Output: Chat removed from MongoDB, 200 OK.

– Result: Pass.

# Conclusion

The Research Paper Summarization project successfully delivers a web-based applica tion that automates the summarization and querying of research papers using a RAG model integrated with Llama 3.3 70B Instruct Turbo Free. Key features include secure user authentication, document upload and processing, AI-driven summarization, interac tive Q&A, and chat management, all supported by a ReactJS frontend, Flask backend, and MongoDB database. The system addresses limitations of traditional summarization methods by ensuring factual accuracy and coherence, making it a valuable tool for re searchers, students, and professionals. Challenges like diverse document formats and API rate limits were overcome through robust text extraction and error handling. The project aligns with KMIT’s mission to promote research-based activities and modern tool usage, as evidenced by its integration of LangChain, FAISS, and advanced LLMs. A demo is available at the provided Google Drive link, showcasing the system’s capabilities.

# Future Enhancements

Future enhancements for KMITPaper include:

• **Performance Optimization:** Implement caching and async processing to handle larger files faster.

• **Additional File Formats:** Support image files formats.

• **Enhanced Table Extraction:** Improve pdfplumber configurations to handle complex tables.

# References

**1.** Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv:2005.11401.

**2.** Nenkova, A., & McKeown, K. (2011). Automatic Summarization. Foundations and Trends in Information Retrieval.

**3.** Zhang, J., et al. (2020). Pegasus: Pre-training with Extracted Gap-sentences for Abstractive Summarization. International Conference on Machine Learning.

**4.** Johnson, J., et al. (2021). Billion-Scale Similarity Search with GPUs. IEEE Transactions on Big Data.

**5. MongoDB Atlas Documentation**: <https://www.mongodb.com/docs/atlas/>.

**6. Flask Documentation**: <https://flask.palletsprojects.com/>.

**7. React Documentation**: <https://reactjs.org/docs/>.

**8. LangChain Documentation:** <https://python.langchain.com/docs/>

# Bibliography

**1.** Gao, Y., et al. (2023). Fine-Tuning Large Language Models for Summarization Tasks. Journal of Artificial Intelligence Research.

**2.** Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing. Prentice Hall.

**3.** Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

**4.** Manning, C. D., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press.