KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

AN AUTONOMOUS INSTITUTION - ACCREDITED BY NAAC WITH 'A' GRADE

Narayanaguda, Hyderabad.

Project Team No.: 33| 4001 |Research Paper Summarization

Team Members

1. Akshaya Batharaju : 23BD1A6646
2. Garige Yashasree : 23BD1A664P
3. Varshini Gopaldas :23BD1A664Q
4. Sampangi Vaishnavi :24BD5A6612
5. Lalitha Vaishnavi :23BD1A058K

Mentors

1. Ms.Kamli
2. Mr. Shankar

CONTENTS

Research Paper Summarization

Using RAG MODEL

INTRODUCTION

Problem Statement:

• Reading and understanding lengthy research papers is time-consuming. Students and researchers often need quick summaries and answers to specific questions.

End User Requirement:

•Quick, accurate, and easy-to-read summaries of lengthy and technical research papers.
•User must be able to choose whether he/she needs short , medium or lengthy summaries based on their time span.
•User must be able to search for the meaning of the word that he finds difficult.
•He must get accurate answers for the questions he poses.

Abstract :

This project focuses on developing an AI-based web application that generates a detailed summary of PDF documents using Large Language Model(LLM) and RAG Model. The main objective is to help users quickly understand lengthy documents without reading them completely. The project uses Python (FastAPI) for backend, Machine Learning models like google gemini pro for text summarization, and ReactJS for frontend. The solution enhances user productivity, especially for students and researchers handling large amounts of content.

Objectives:

•Reduce time spent on initial research review.
•Enhance understanding of complex topics through concise summaries.
•Offer tailored summarization options (short, medium, long).
•Simplify exploration of unknown terms via integrated tools.

Technology Stack:

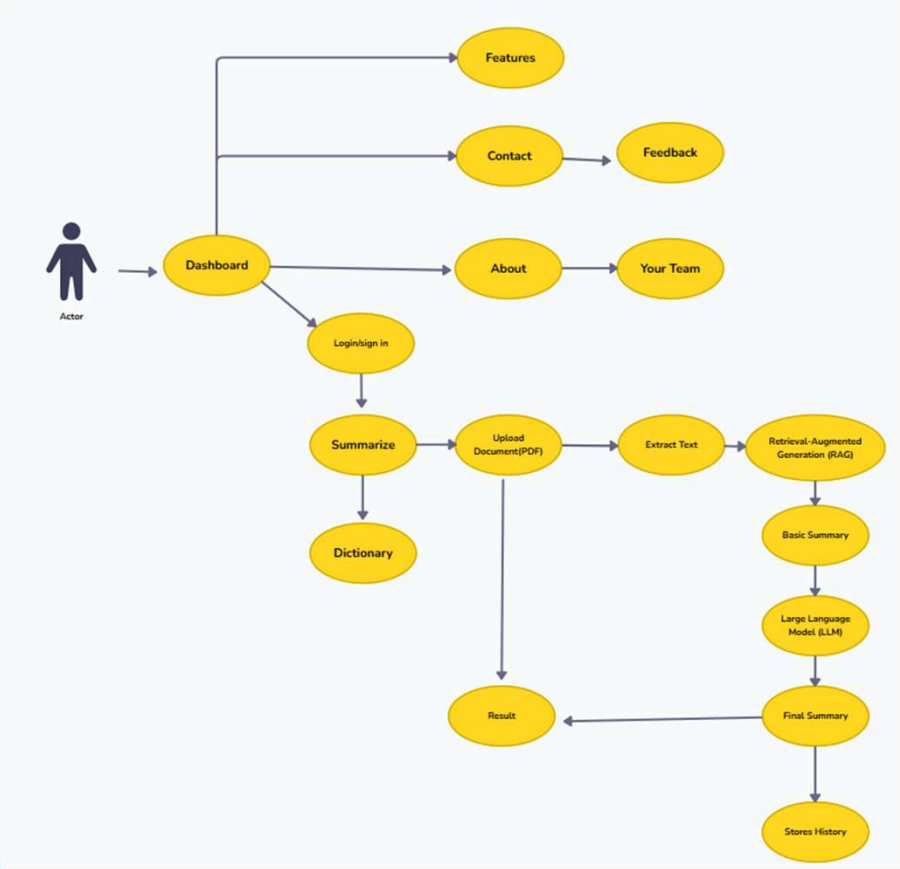
•Artificial Intelligence & Machine Learning
FIASS as Vector Database
Retrieval-Augmented Generation (RAG)
Large Language Models (LLMs) – Google Gemini Pro

REQUIREMENT

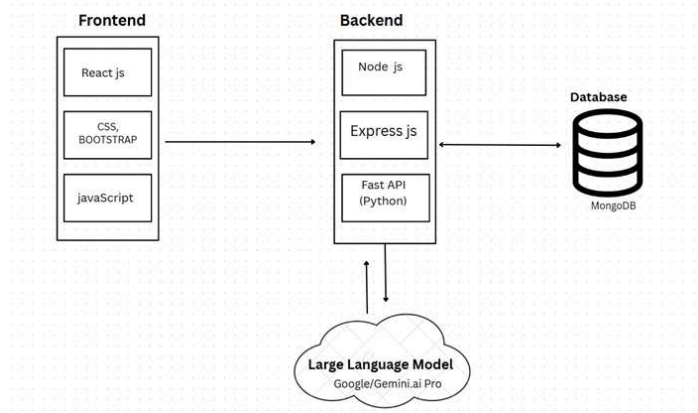
Outcome:
A web-based, intelligent summarization tool for academic content that simplifies and accelerates the research process.

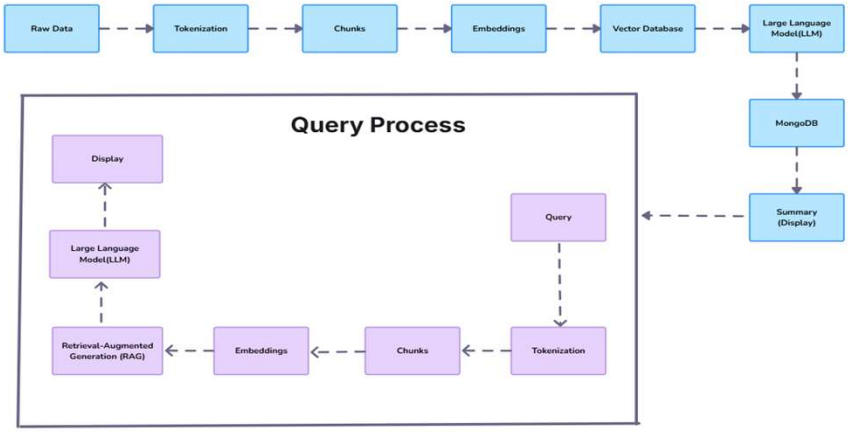
Key Features:
- **Automated summarization of uploaded research papers (PDF, DOC):**
Enables users to upload various document formats and receive instant, AI-generated summaries.
- **Adjustable summary length (short / medium / detailed):**
Allows users to choose the depth of the summary based on their needs—quick scan or deep understanding.
- **Dark and light mode themes:**
Enhances user comfort and accessibility through customizable display preferences.
- **Built-in academic dictionary :**
Offers clear definitions of technical terms to aid comprehension, especially for interdisciplinary readers.
- **User feedback submission for improvement:**
Collects input from users to refine summaries and enhance future platform updates.
- **Summarization history :**
History of the summaries is also stored..
- **AI-powered chatbot for Q&A based on uploaded content:**
Provides interactive assistance and answers user queries derived directly from the research paper content.
- **Text to Speech :**
The generated summary is converted to speech.

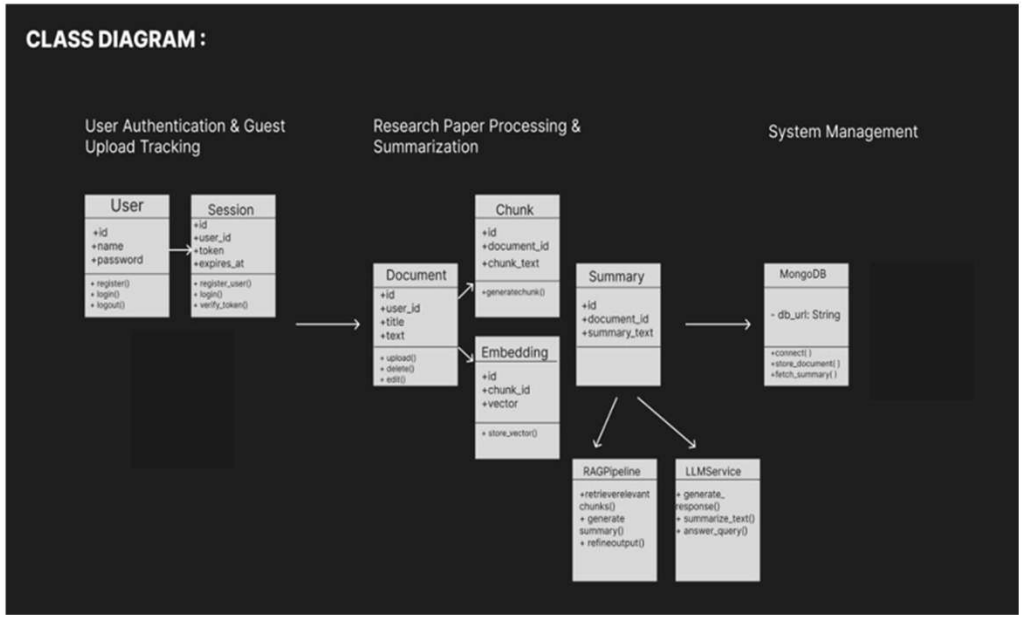
Use Case :



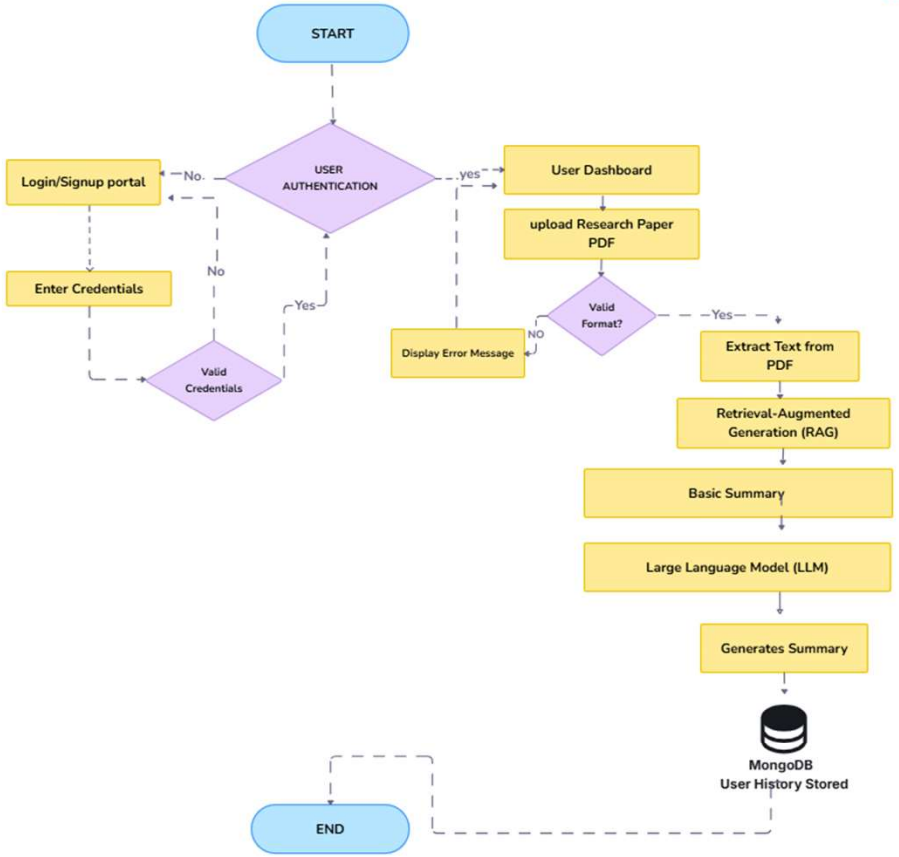
Architecture :





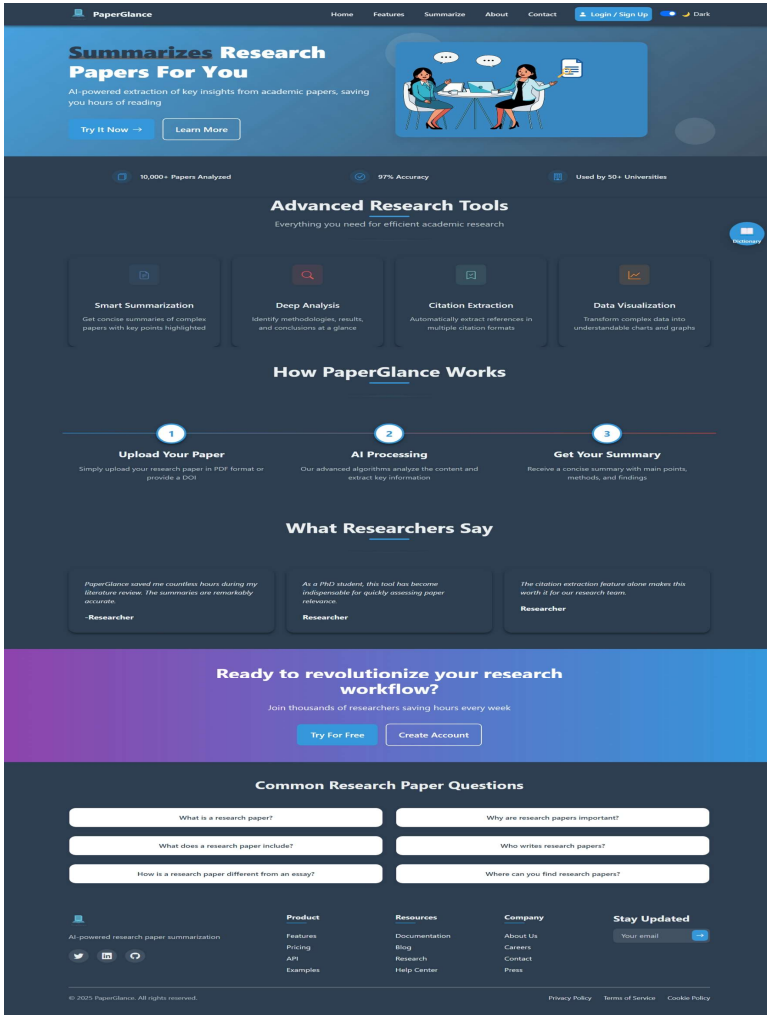


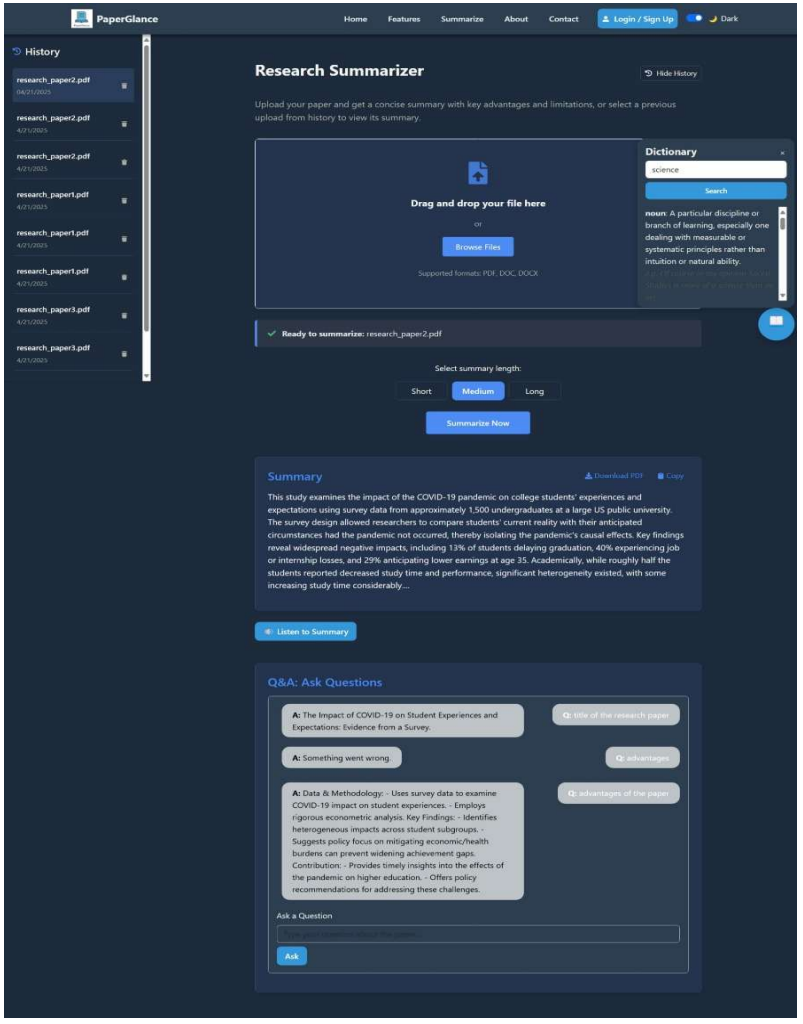
FlowChart:



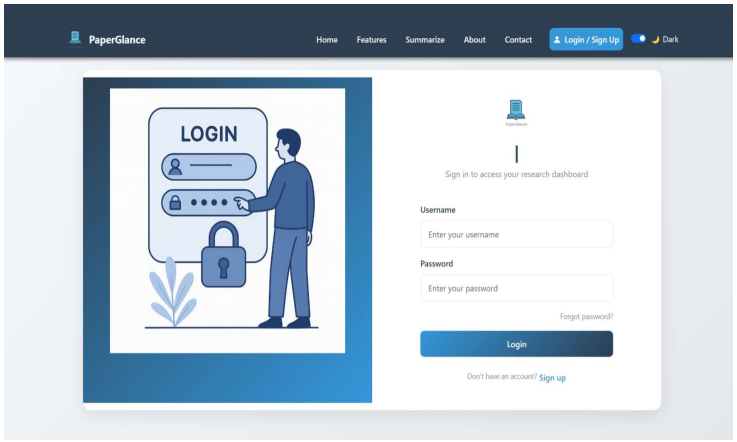
DataBase Design:

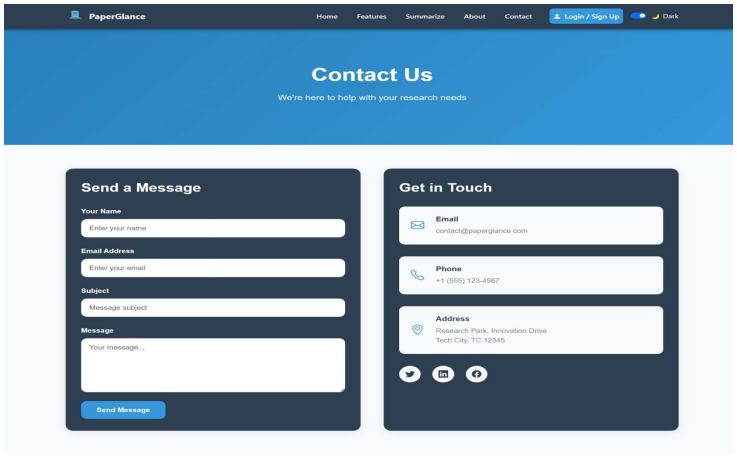
DEVELOPMENT






PaperGlance
Home
Features
Summarize
About
Contact
Login / Sign Up
Dark
About PaperGlance
Revolutionizing Academic Research Through AI
Our Mission
At PaperGlance, we're transforming how researchers interact with scholarly content. Our AI-driven platform delivers concise, accurate summaries of complex research papers, accelerating your academic workflow.
Why Choose PaperGlance
Smart Summarization
AI-powered summaries capture the essence of complex papers in minutes
Academic Integrity
Algorithms designed to maintain scholarly rigor and accuracy
Critical Analysis
Highlight key strengths and limitations for balanced evaluation
Community Driven
Developed with continuous feedback from researchers worldwide
Our Team
VG
Varshini Gopaldas
Research Specialist
LV
Laitha Vaishnavi
Research Specialist
YG
YashaSree Garige
Research Specialist
VS
Vaishnavi Sampangi
Research Specialist
AB
Akshaya Batharaju
Research Specialist
Key Features
Instant paper summarization
Citation pattern analysis
Key term identification
Research trend tracking
Transform Your Research Process
Start Free Trial

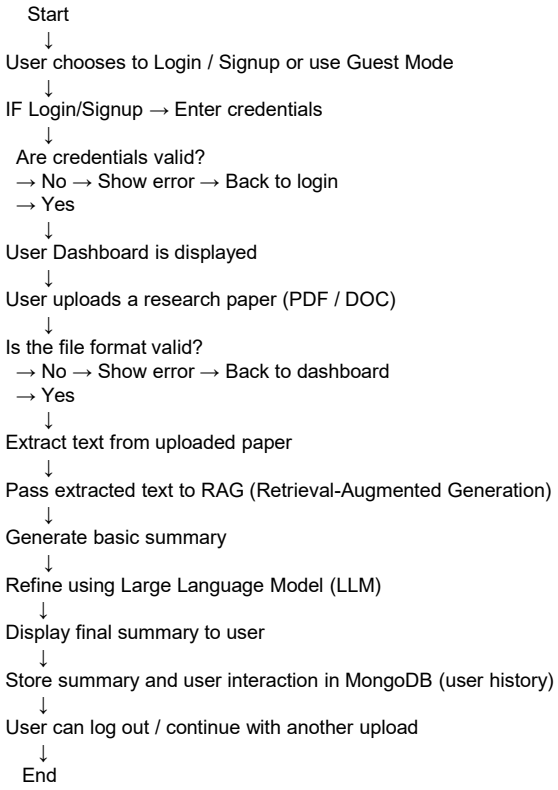

PaperGlance
Home
Features
Summarize
About
Contact
Login / Sign Up
Dark
LOGIN
Sign in to access your research dashboard
Username
Enter your username
Password
Enter your password
Forgot password?
Login
Don't have an account? Sign up


PaperGlance
Home
Features
Summarize
About
Contact
Login / Sign Up
Dark
Contact Us
We're here to help with your research needs
Send a Message
Your Name
Enter your name
Email Address
Enter your email
Subject
Message subject
Message
Your message...
Send Message
Get in Touch
Email
contact@paperglance.com
Phone
+1 (555) 123-4567
Address
Research Park, Innovation Drive
Tech City, TC 12345

Tools and Frameworks Used:

- **NLP & ML:** Google/Gemini pro 1.5
- **Frontend:** React, Bootstrap, CSS Modules
- **Backend:** FastAPI
- **Database:** MongoDB
- **Others:** Git, Postman, Google Colab, Figma (for UI prototypes)

Workflow:



Frontend Pseudo code

api.js

```
SET API_URL = "http://localhost:8000"

FUNCTION uploadFile(file):
    CREATE formData with file
    TRY:
        POST to API_URL + "/upload" with formData
        RETURN response.data
    CATCH:
        LOG error
        THROW "File upload failed"

FUNCTION getSummary(query):
    TRY:
        POST to API_URL + "/summarize" with query
        RETURN response.data
    CATCH:
        LOG error
        THROW "Summary generation failed"
```

SummarizePage.js

```
INITIALIZE state:
    isDragging, fileName, file, showHistory, loading,
summaryResult, error,
    summaryLength = "medium", fileUploaded, historyItems,
copiedText, selectedHistoryItem

ON MOUNT:
    FETCH history from "/summaries/"

FUNCTION fetchHistory():
    GET "/summaries/"
    MAP response to historyItems with id, name, date,
summary, advantages, disadvantages, insights

FUNCTION handleHistoryItemClick(item):
    SET selectedHistoryItem = item
    SET summaryResult = item data
    CLEAR error, file, fileName, fileUploaded

FUNCTION handleDownloadSummary():
    POST summaryResult.summary to "/download-summary/"
    CREATE and TRIGGER PDF download

FUNCTION handleDeleteHistoryItem(id):
    DELETE "/delete-summary/" + id
    REMOVE id from historyItems
    IF selected, CLEAR selectedHistoryItem, summaryResult

FUNCTION handleDragOver():
    SET isDragging = true

FUNCTION handleDragLeave():
    SET isDragging = false

FUNCTION handleDrop(event):
    SET file, fileName from dropped file
    CLEAR summaryResult, error, selectedHistoryItem
    SET fileUploaded = true
    CALL handleUpload(file)

FUNCTION handleFileChange(event):
    IF file type valid (PDF, DOC, DOCX):
        SET file, fileName
        CLEAR summaryResult, error,
selectedHistoryItem
        SET fileUploaded = true
        CALL handleUpload(file)
    ELSE:
        SET error = "Unsupported file type"

FUNCTION handleUpload(file):
    POST file to "/upload/"
    ADD new item to historyItems

FUNCTION handleSummarize():
    IF no file, RETURN
    SET loading = true
    POST file, summaryLength to "/summarize"
    IF success:
        SET summaryResult with response data
        CLEAR selectedHistoryItem
    ELSE:
        SET error
    SET loading = false

FUNCTION handleSummaryLengthChange(length):
    SET summaryLength = length

FUNCTION handleCopyText(text):
    COPY text to clipboard
    SET copiedText = true, RESET after 2s

RENDER:
    SHOW sidebar (if showHistory) with historyItems
    SHOW main content:
        File upload (if no selectedHistoryItem)
        Summary length buttons (if fileUploaded)
        Summarize button (if no selectedHistoryItem)
        Error message (if error)
        Summary output (if summaryResult) with
download, copy, TextToVoice
        ChatBox (if summaryResult.summary)
        DictionaryWidget
```

Backend Pseudo code

llm_utils.py

```
SET MODEL = "gemini-1.5-pro"

FUNCTION get_valid_headers():
    FOR each api_key in API_KEYS:
        TRY:
            POST test request to Gemini API with api_key
            IF status == 200:
                RETURN api_key
        CATCH:
            LOG error
    THROW "All API keys failed"

FUNCTION clean_summary(summary):
    REMOVE markdown headers, bold, italic
    NORMALIZE bullet points to "- "
    REMOVE HTML tags, chatty phrases
    NORMALIZE whitespace
    RETURN cleaned summary

FUNCTION summarize_with_llm(chunks,
summary_length="medium"):
    SET prompt based on summary_length:
        short: 1-2 sentences, use first 5 chunks
        medium: 2-3 paragraphs, use first 10 chunks
        long: 6-8 paragraphs with headings, use all chunks
        default: medium
    CREATE payload with prompt
    GET valid api_key
    TRY:
        POST to Gemini API with payload
        EXTRACT raw_summary
        RETURN
format_summary(clean_summary(raw_summary),
summary_length)
    CATCH:
        LOG error
        THROW "Error summarizing"
```

```
FUNCTION ask_question(question, top_chunks):
    SET prompt:
        IF question contains "title", "author", etc.:
            Brief answer (max 50 words)
        ELSE:
            Concise bullet points with headings (max 100 words)
    CREATE payload with prompt
    GET valid api_key
    TRY:
        POST to Gemini API with payload
        RETURN format_summary(clean_summary(raw_answer))
    CATCH:
        LOG error
        THROW "Error answering"

FUNCTION format_summary(summary, summary_length="medium"):
    IF summary_length == "short":
        TRUNCATE to 100 words
    SPLIT into sections
    FORMAT sections with headings and normalized bullet points
    RETURN formatted summary
```

rag_utils.py

```
SET CHUNK_SIZE = 150
SET CHUNK_OVERLAP = 25
INITIALIZE embedder = SentenceTransformer('all-MiniLM-L6-v2')

FUNCTION chunk_text(text):
    SPLIT text into words
    RETURN chunks of CHUNK_SIZE with CHUNK_OVERLAP

FUNCTION process_pdf(pdf_path):
    OPEN PDF with fitz
    EXTRACT text from all pages
    CREATE chunks using chunk_text
    GENERATE embeddings with embedder
    CREATE FAISS index with embeddings
    RETURN chunks, index

FUNCTION get_top_chunks(query, chunks, index, top_k=5):
    ENCODE query with embedder
    SEARCH index for top_k closest chunks
    RETURN corresponding chunks
```

main.py

```
INITIALIZE FastAPI app
CONFIGURE CORS, logging, wkhtmltopdf
SET UPLOAD_DIR = "uploads"
CONNECT to MongoDB with collections: papers, summaries

ROUTE POST /upload:
    SAVE uploaded file to UPLOAD_DIR
    PROCESS PDF to get chunks, index
    GENERATE summary with summarize_with_llm
    SAVE index to pickle file
    INSERT paper document to papers_collection
    INSERT summary document to summaries_collection
    RETURN paper_id, filename, summary, date

ROUTE POST /summarize:
    SAVE uploaded file
    PROCESS PDF to get chunks, index
    GENERATE summary with summarize_with_llm(summary_length)
    SAVE index to pickle file
    INSERT paper and summary documents
    RETURN paper_id, summary, empty advantages/disadvantages/insights

ROUTE GET /summaries:
    FETCH up to limit summaries, sorted by created_at
    FOR each summary:
        ADD paper details (filename, upload_date)
        CONVERT IDs to strings
    RETURN summaries

ROUTE POST /chat:
    VALIDATE question, paper_id
    IF no paper_id, USE most recent paper
    FETCH paper by paper_id
    IF no chunks/index, PROCESS PDF
    GET top 3 chunks for question
    GENERATE answer with ask_question
    RETURN answer

ROUTE POST /download-summary:
    RECEIVE summary_text
    CREATE HTML with summary_text
    CONVERT to PDF using pdfkit
    RETURN PDF file

ROUTE DELETE /delete-summary/{summary_id}:
    DELETE summary by summary_id
    IF not found, THROW 404
    RETURN success message

ROUTE POST /api/define:
    CORRECT word spelling
    FETCH definition from dictionary API
    IF found:
        RETURN word, suggested correction, definitions
    ELSE:
        RETURN 404 error
```

PROJECT OUTCOME

Achievements

•**Developed :**Delivered a working version of the application that demonstrates core features, user flow.

•**Successfully handled summarization of research Paper across domains**

Achieved reliable and context-aware summarization for a wide range of topics including computer science, engineering, and social sciences.

• **Integrated helpful utilities** .

- Helps researchers, students, and professionals quickly understand key insights, saving time and boosting efficiency.
- Personalized User Experience :, history tracking, saved summaries, and customization options like summary length and voice mode.
- Enhanced Accessibility : Voice input and text-to-speech features improve usability for differently-abled users and those preferring audio output.
- Multilingual Document Summarization : Allows uploading papers in any language (e.g., Telugu) and generates summaries in English, enabling cross-language understanding of academic content.
- Interactive Learning Environment : AI Chat Board supports conversation-based queries, deeper insights, definitions, and explanation of paper content.
- User Feedback Integration :Feedback system helps gather user input to improve system quality and user satisfaction over time.

Key Challenges

• **Preserving technical accuracy**

Summarizing complex academic language without losing key details, especially in research or formula-based papers.

• **Speed optimization for large PDFs and documents**

Ensuring quick responses and fast loading, even with documents that have a lot of pages.

•**Designing an intuitive yet powerful UI**

Designing a simple, easy-to-use interface while including advanced features like voice-to-text from AI chat bot, and History saving for all users.

CONCLUSION

A fully functional, web-based application has been developed to summarize research papers efficiently. The platform leverages **Large Language Models (LLMs)** and **Retrieval-Augmented Generation (RAG)** techniques in the backend to generate high-quality, context-aware summaries. The frontend is built using **React**, providing a clean, responsive, and user-friendly interface. The system supports various features like adjustable summary length, dark/light mode, an integrated dictionary, chatbot assistance, and user history tracking — making it a powerful tool for students, researchers, and professionals alike.
It efficiently summarizes research papers and provides a user-friendly interface with rich academic support features.

New Skills Learned:

- Implementation of LLMs and Retrieval-Augmented Generation.
- UX design tailored for research-heavy tools.
- Working with frontend-backend integration and secure authentication.
- Managing user data securely with database systems.

Thank you!