Progress Presentation – 01

Research Topic

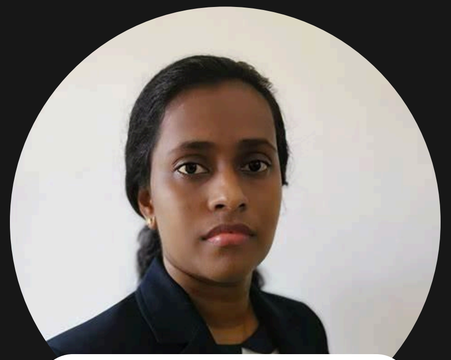
Adaptive Learner Centric LMS

Enhanced Student Engagement and Personalised Learning Experiences

SLIIT FACULTY OF COMPUTING

24-25J-112

Who we are



Supervisor

Dr. Kalpani Manathunga

Co-Supervisor

Ms. Thilini Jayalath

IT21251900

Rajapaksha R.M.S.D.

IT21302862

Sri Samadhi L.A.S.S

IT21302862

Peiris A.R.D.

IT21360428

Monali G.M.N.

Agenda

Problem and background

Topic and objectives

Project and architecture

Software engineering aspect

Individual Components

Background

Which tools do you typically use when preparing to study? (Select all that apply) *

- [ ] LMS to download notes and past papers
- [ ] PDF viewer
- [ ] Word processor
- [ ] ChatGPT or other AI tools
- [ ] Timer or study planner
- [ ] Diagram drawing tools
- [ ] Music streaming tool (Background Music)

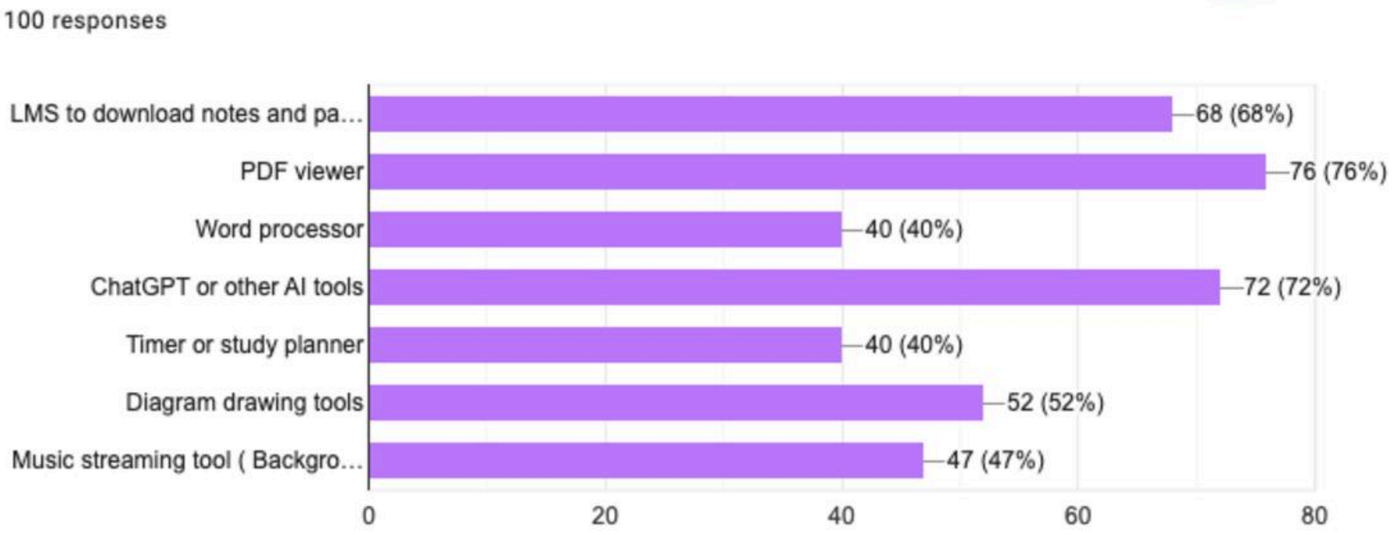


Figure 2.1
Results of a survey conducted around 100 university / collage students who daily uses LMSs in 2024

Problem

What challenges do you face with your current study routine? (Select all that apply) *

- [ ] Takes a long time to get started and open up all tools (eg: pdfs, note taking apps, chat bots)
- [ ] Unorganized and poorly named documents in LMS
- [ ] No customizable layouts or background options when reading lecture materials
- [ ] Can't change fonts on a reading materials
- [ ] Cant edit, highlight or save your own version of the learning materials on LMS.
- [ ] Frequent distractions when switching between applications while studying.
- [ ] Disliked layouts and UI of the LMS.
- [ ] Difficulty in organizing study materials (Have to do it manually on a local machine)
- [ ] Not student-centric (Students can only read and download materials and use forums)
- [ ] Need to use multiple tabs, tools or screens when studying.

Figure 2.2
Results of a survey conducted around 100 university / collage students who daily uses LMSs in 2024

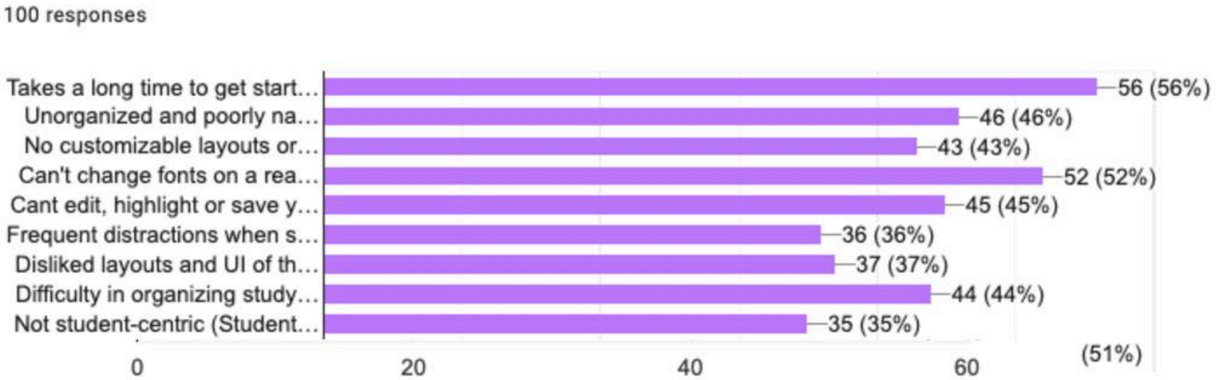
Waste of time

Disengagement

Procrastination

Unorganized Content

Lack of Customizability



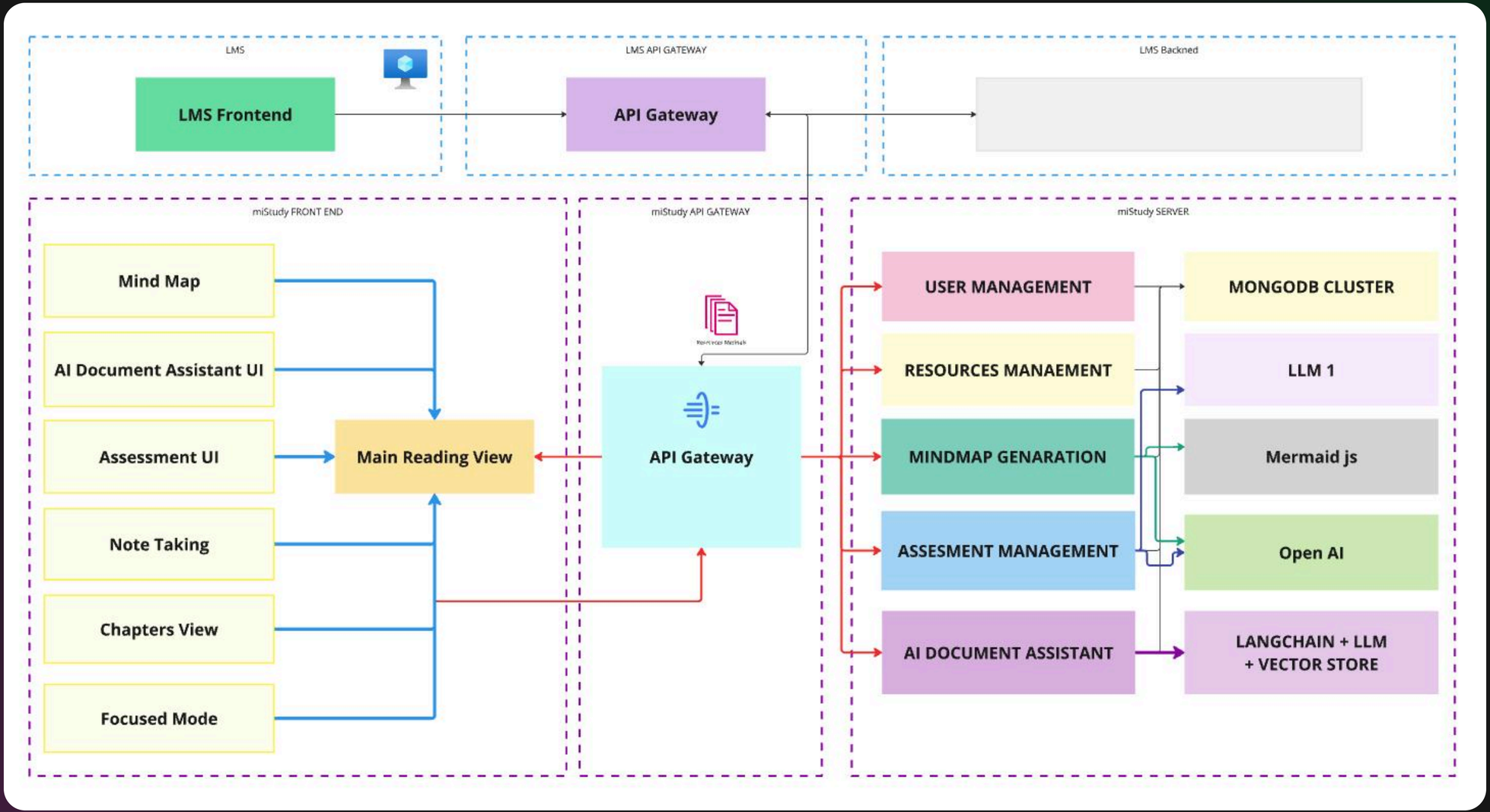
Research question

“How does the implementation of an Adaptive Learner-Centric Learning Management System (LMS) impact student engagement, autonomy, and motivation compared to traditional LMS implementations?”

Objectives

- Enhance Student Engagement
- Personalize Learning Experiences
- Improve Communication Features
- Shift to Learner-Centric Approach
- Utilize Interactive, Adaptive Content
- Create Supportive Educational Environment

Architecture



Idea and commercialisation

A Cloud based SASS Application

Target Market

Any Individual who are looking for application/platform to manage organise their documents and who interacts with a lot of resources.

A plugin interface

Target Market

Universities and higher education institutions with already using LMS, looking to enhance student engagement and academic performance through innovative learning technologies.

Revenue Model

Subscription-Based Plans

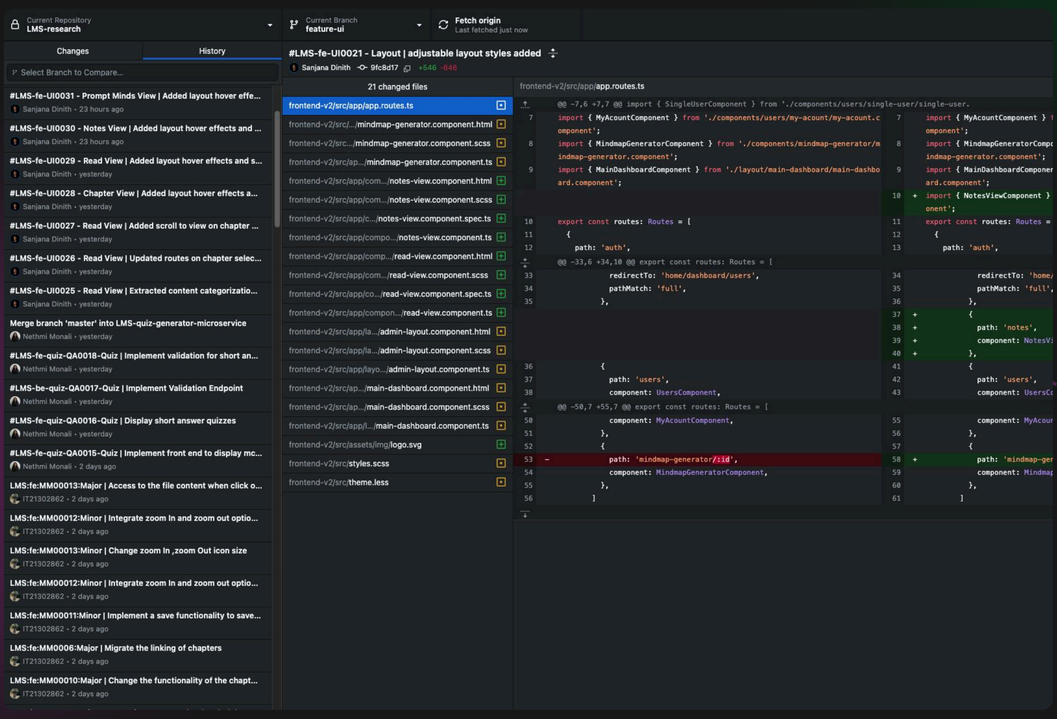
Monthly or annual subscription fees for institutions, based on the number of users and selected feature sets.

Freemium Model

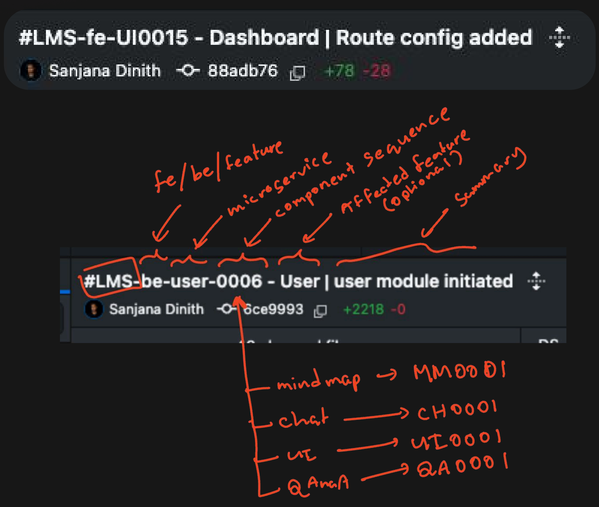
Offering a basic version of the LMS for free, with premium features available through paid subscriptions.

Software engineering aspect

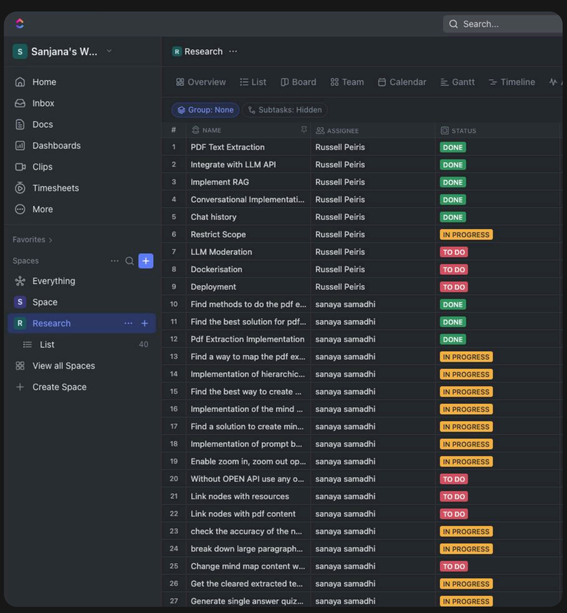
GitHub for version Control

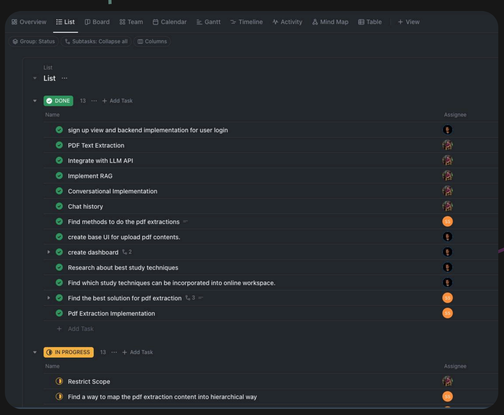


Commit Structure



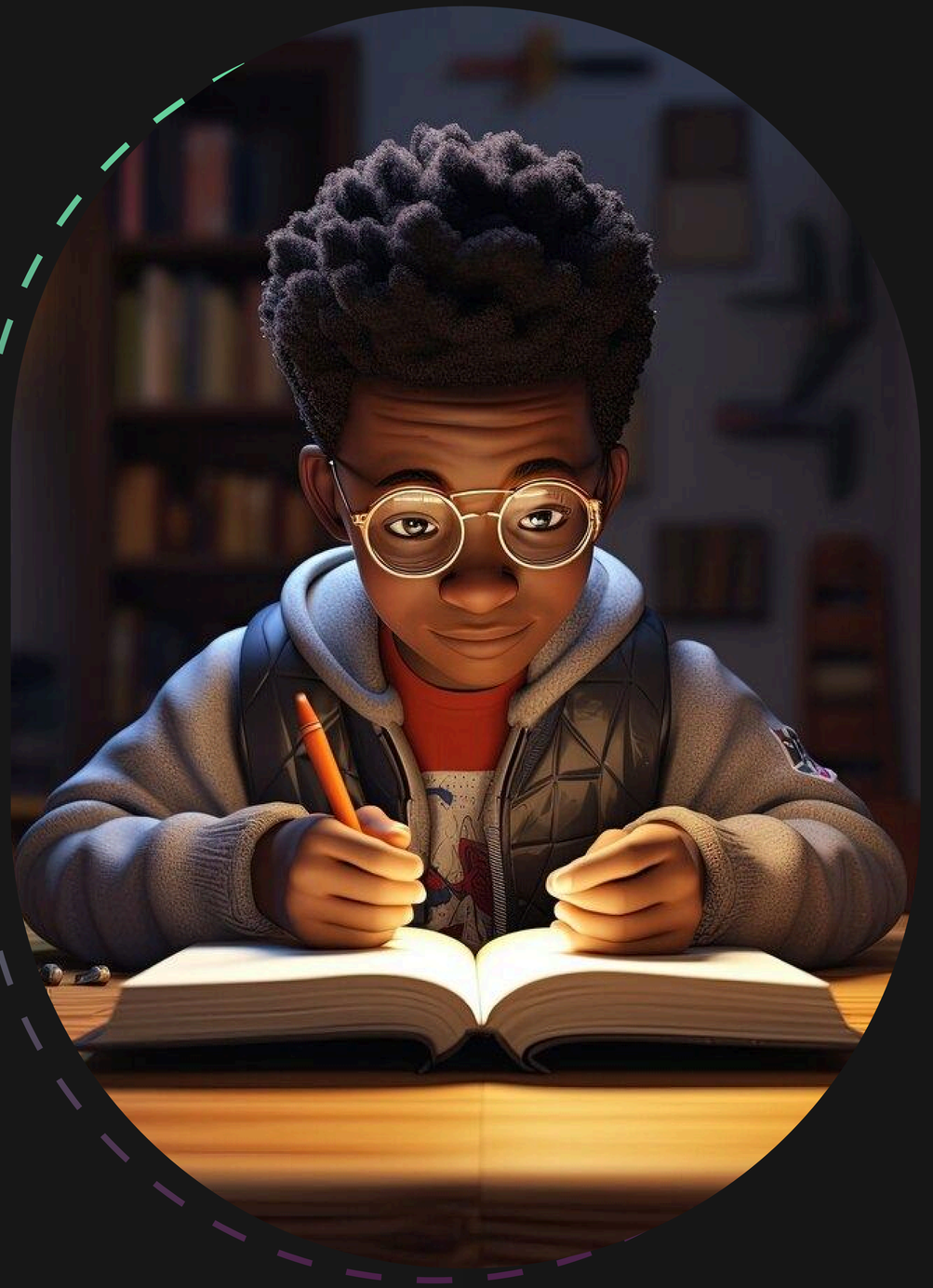
Project Management _ Clickup





IT21251900 – Rajapaksha R.M.S.D

Keep Students Engaged in a Learning Environment



Implement all rounded interface.

Incorporate study techniques.

Research Areas

Human computer interaction

User Interface and Experience Design

Educational Technology

Learning Techniques / E-Learning Platforms

Behavioral Psychology

Motivation and Engagement / Learning Behaviors

Software Engineering

Optimization, Scalability and Performance

Cognitive Science

Memory and Retention/Cognitive Load Management

Objectives

Incorporate study techniques

- To identify the best study techniques that can improve learning.
- To determine which of these techniques can be incorporated into an online learning environment and how to implement them effectively.
- To create a personalised interaction experience within the LMS.

Incorporate all the tools into one interface effectively.

- To identify the most essential tools that students need when studying.
- To present all features in a less distracting and more attractive manner.
- To optimise the system for balanced client-side load and performance.

Problems that i cover

- [x] Takes a long time to get started and open up all tools (eg: pdfs, note taking apps, chat bots)
- [x] Unorganized and poorly named documents in LMS
- [x] No customizable layouts or background options when reading lecture materials
- [x] Can't change fonts on a reading materials
- [x] Cant edit, highlight or save your own version of the learning materials on LMS.
- [x] Frequent distractions when switching between applications while studying.
- [x] Disliked layouts and UI of the LMS.
- [x] Difficulty in organizing study materials (Have to do it manually on a local machine)
- [x] Not student-centric (Students can only read and download materials and use forums)
- [x] Need to use multiple tabs, tools or screens when studying.

Objectives Breakdown

Incorporate study techniques

To identify the best study techniques that can improve learning.

To determine which of these techniques can be incorporated into an online learning environment and how to implement them effectively.

To create a personalised interaction experience within the LMS.

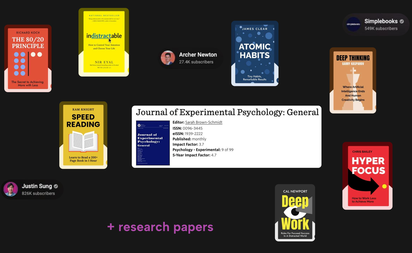
Initial Plan

	Techniques used	Proven	Personalized	Tested Online	Used in LMS
Research 01 [2]	Pomodoro + Forest app	✓	x	✓	✓
Research 02 [3]	Pomodoro	✓	x	✓	x
Research 03 [4]	Pomodoro	✓	x	✓	x
Research 04 [5]	Spaced Repetition	✓	x	x	x
Research 05 [6]	Feynman	✓	x	x	x
Research 06 [7]	Background Music	✓	x	✓	x
Article [8]	Flowtime	✓	✓	✓	x
Video [9]	Flowtime + pomodoro	x	✓	x	x

Keywords

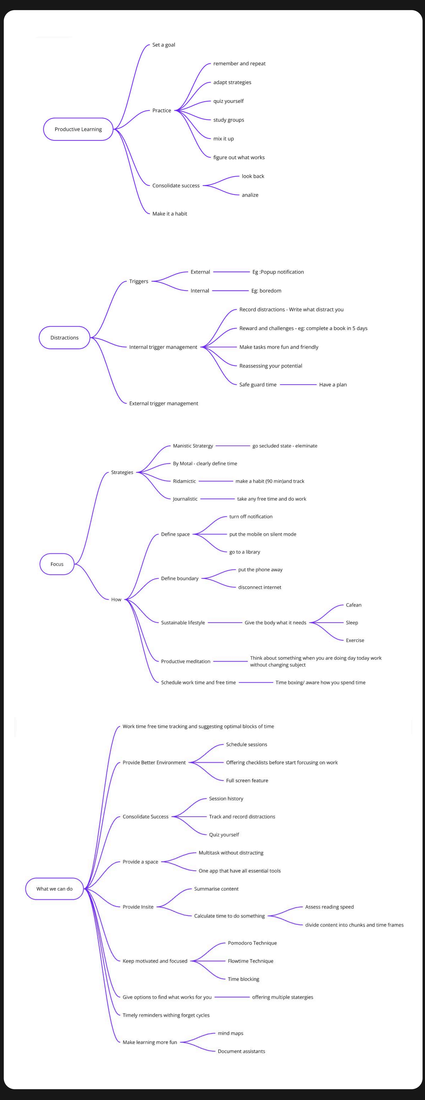
Distractions
Motivation
Focus
Engagement
Procrastination
Time Management
Productivity
Memory
Habits

Sources



+ research papers

Research Flow



Identified Techniques

Progressive Pomodoro

Based on Pomodoro

- Traditional Pomodoro – (25-min work/5-min break cycles),
- Starting with a longer work period
- Gradually shortening both work and break durations
- Helps maximise productivity while preventing burnout
- 60/20 40/10 20/5

Flomodoro

Based on Pomodoro and Flowtime

- Work for as long as you can
- Then calculate your break based on the time spent working
- Flexible focus and rest strategy
- Prioritises maintaining deep work states (flow)

IT22125900 – Rajapaksha R.M.S.D

Objectives Breakdown

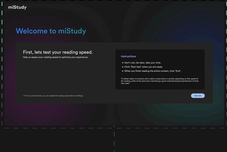
Incorporate study techniques

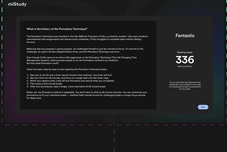
- To **identify the best study techniques** that can improve learning.
- To **determine which of these techniques can be incorporated** into an online learning environment and how to implement them effectively.
- To **create a personalised interaction experience** within the LMS.

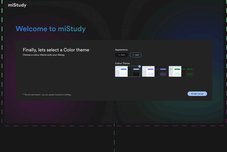
Incorporate all the tools into one interface effectively.

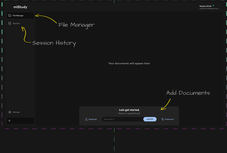
- To **identify the most essential tools** that students need when studying.
- To **present all features in a less distracting and more attractive manner.**
- To **optimise the system for balanced client-side load and performance.**

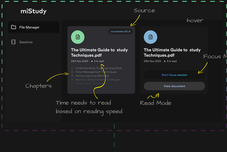
User Flow
&
User Interface

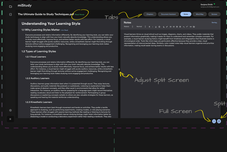


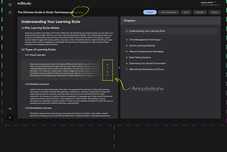


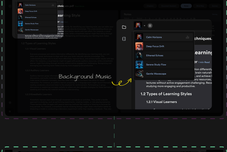


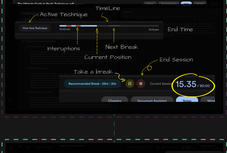


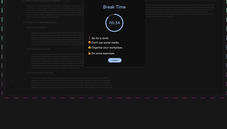












Progress

Whats done.

- Identify the best study techniques.
- Determine which of these techniques can be incorporated into an online learning environment.
- How to implement them effectively.
- Design Application UI
- Dev – Application structure design
- Dev – User Module Added
- Dev – Base application developement

Whats ToDo.

- Dev – Study technique related actions integrations
- Dev – User session management
- Dev – Document history management
- Dev – Customisation and personalizations

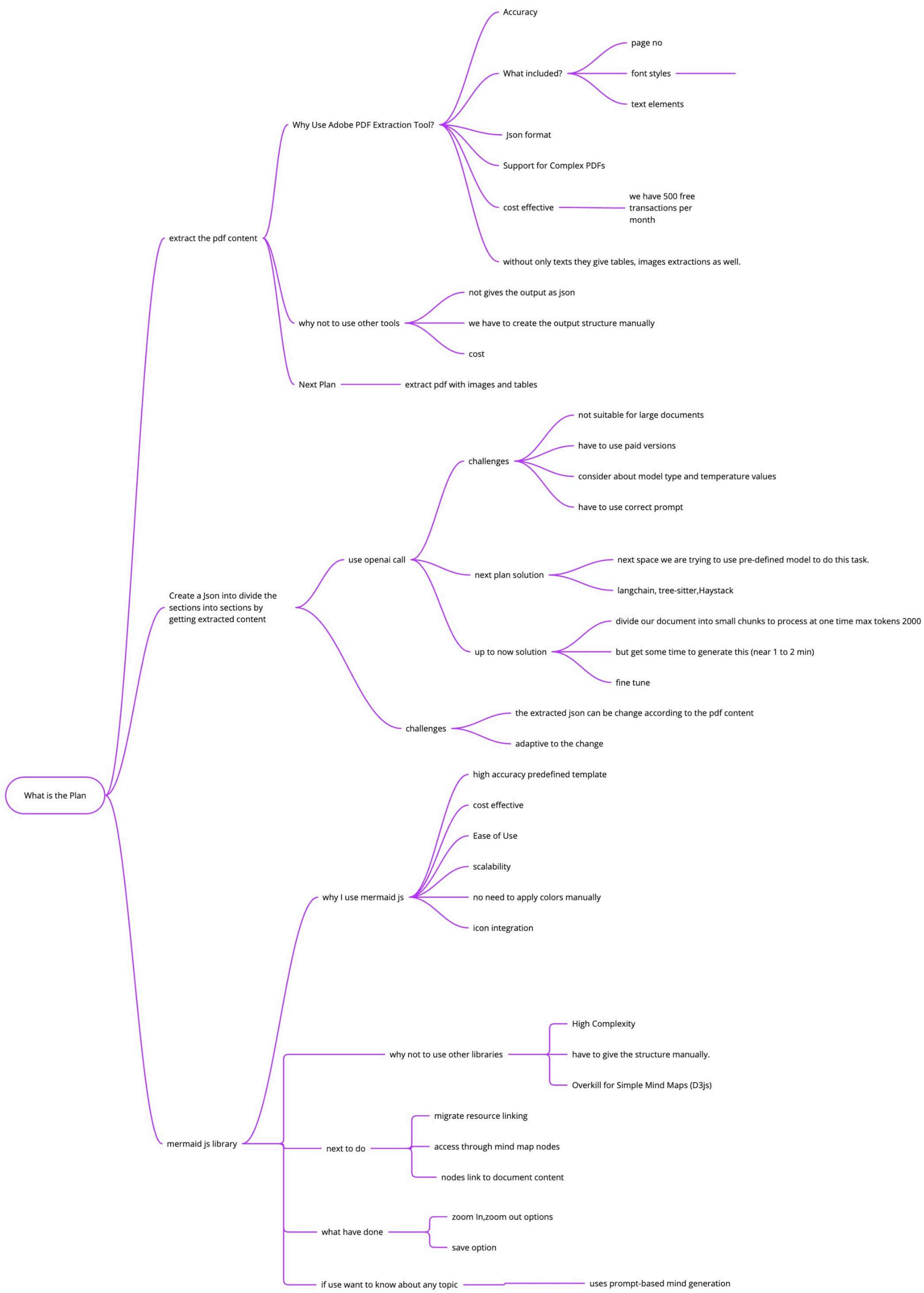
IT21302862 – Sri Samadhi L.A.S.S

Adaptive Mind Map Generation.

Mind maps tailored to individual learners.

Provide additional resource via mind map.

Current Workflow



Optimising PDF Content Extraction and Visualisation

Current Workflow Overview

- **Extract content from PDF using Adobe PDF Extraction Tool.**

Why Use Adobe PDF Extraction Tool?

- **High accuracy in extracting text, tables, and structured data.**
- **Support for Complex PDFs**
- **Handles scanned documents and complex layouts effectively.**
- **Cost effective – Give 500 free transactions per month.**
- **Provide JSON format easy way to get all the things not only the text we can get page no, font styles all the things in structured way.**


```
24      "elements": [
56          {
57              "Bounds": [
60                  236.5172882080078,
61                  660.6699981689453
62              ],
63              "Font": {
64                  "alt_family_name": "Calibri",
65                  "embedded": true,
66                  "encoding": "WinAnsiEncoding",
67                  "family_name": "Calibri",
68                  "font_type": "TrueType",
69                  "italic": false,
70                  "monospaced": false,
71                  "name": "BCDEEE+Calibri-Bold",
72                  "subset": true,
73                  "weight": 700
74              },
75              "HasClip": true,
76              "Lang": "en",
77              "ObjectID": 448,
78              "Page": 0,
79              "Path": "//Document/H1",
80              "Text": "1.1 What is Networking? ",
81              "TextSize": 14.039993286132812,
82              "attributes": {
83                  "LineHeight": 16.875
84              }
85          },
86          {
87              "Bounds": [
88                  90.02400207519531,
```

```
24      "elements": [
3053        {
3078        },
3079        {
3080            "Bounds": [
3081                144.02000427246094,
3082                223.07911682128906,
3083                495.71343994140625,
3084                234.11912536621094
3085            ],
3086            "Font": {
3087                "alt_family_name": "Cambria",
3088                "embedded": true,
3089                "encoding": "WinAnsiEncoding",
3090                "family_name": "Cambria",
3091                "font_type": "TrueType",
3092                "italic": false,
3093                "monospaced": false,
3094                "name": "BCDGEE+Cambria",
3095                "subset": true,
3096                "weight": 400
3097            },
3098            "HasClip": true,
3099            "ObjectID": 449,
3100            "Page": 3,
3101            "Path": "//Document/Sect[3]/Sect[2]/L[2]/LI[2]/LBody",
3102            "Text": "Scalability: New devices can be added seamlessly with
3103            "TextSize": 11.039993286132812
3104        }
3105    ],
3106    "pages": [
3107        {
```


Why not to use other tools?

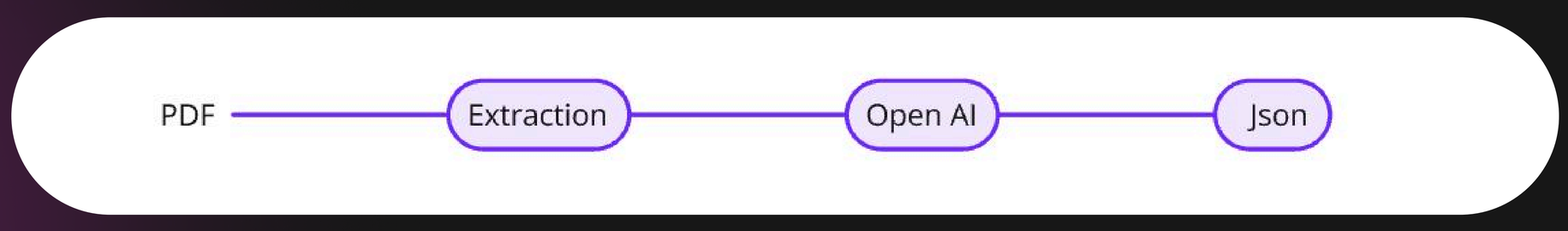
- **Want to create the structure manually ,**
- **Costly,**
- **Extract only the text – can't get the fonts and pages and text formats.**
- **Lack of Advanced Features – not extract tables contents and image contents.**
- **eg: PDF js, PyPDF2**

Saving hierarchical Json into Database.

- Extract content from PDFs using Adobe PDF Extraction Tool.
- Use OpenAI API to process and map the extracted content into a hierarchical JSON structure..

Outcome

- Works well for small documents.
- Challenges emerge with large documents.

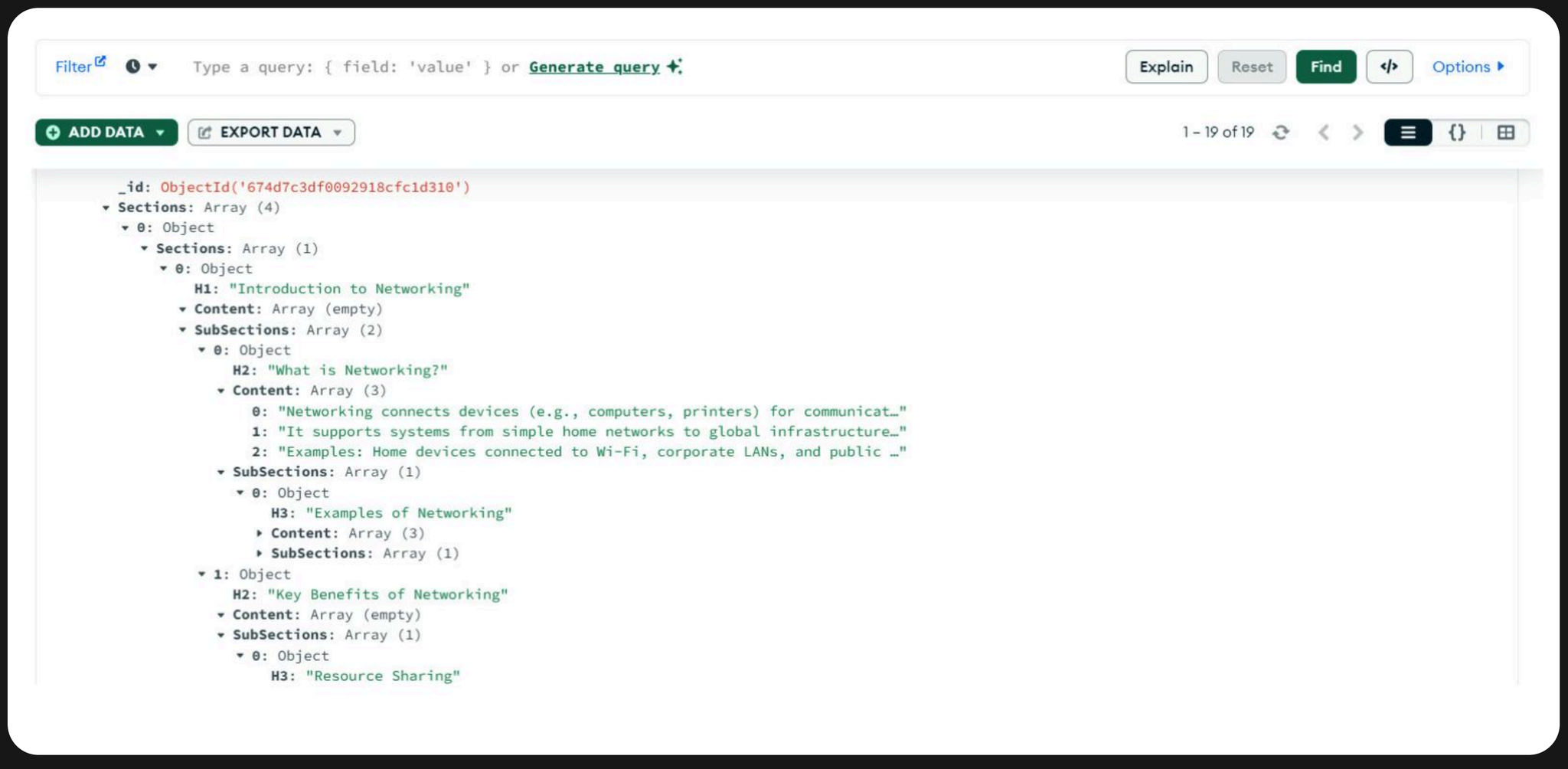


Why Current Approach Struggles

- Scalability Issues
- OpenAI's token limits make it hard to process large documents in a single call.
- Splitting documents into smaller chunks adds complexity.
- High Costs – Processing large documents via multiple API calls increases expenses.
- Performance Bottlenecks – Increased latency and time required for chunk processing. Implement batch processing for smaller chunks. but still it gets some time consuming

Database Structure

After diving it to chunk sizes how it looks like :



Next Steps for Scalability

- Integrate Alternative Tools
- Evaluate tools like LangChain or Haystack for hierarchical content mapping.
- Use local processing tools for pre-structuring the data before sending it to OpenAI.
- Optimise OpenAI Usage
- Fine-tune OpenAI models for domain-specific tasks to reduce token usage.

Mind map generation

Why Mermaid.js?

Ease of use

Simple syntax for defining diagrams in Markdown.
No need for complex design tools or graphic software.

Integration-Friendly

Works seamlessly with web technologies and integrates well into JavaScript/Typescript-based applications.

Compatible with tools like Markdown editors and static site generators.

Customizability

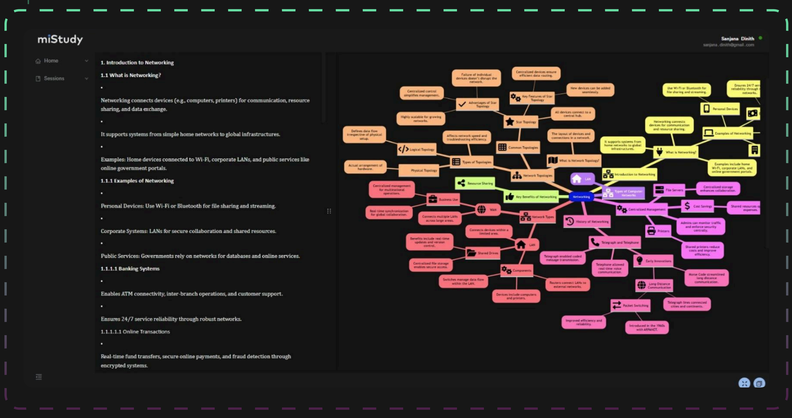
Supports styling and theming options for diagrams.
Allows customisation to align with project requirements.

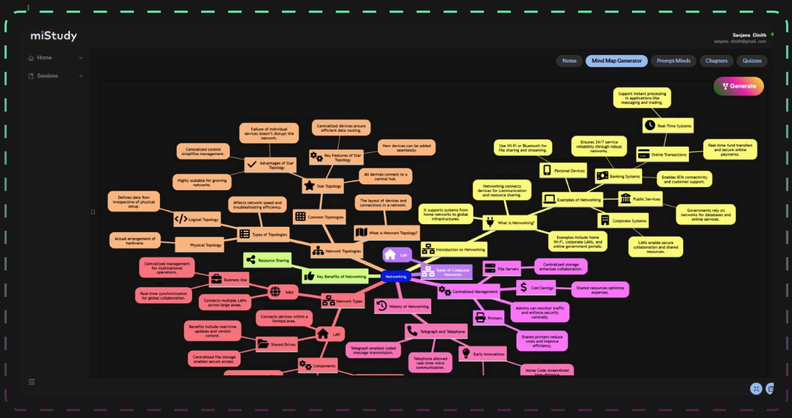
Perfromance

Lightweight and fast rendering compared to other visualisation libraries.

Accessibility

Can use with material UI icons for the nodes.
Free and open-source, avoiding licensing issues or subscription costs.





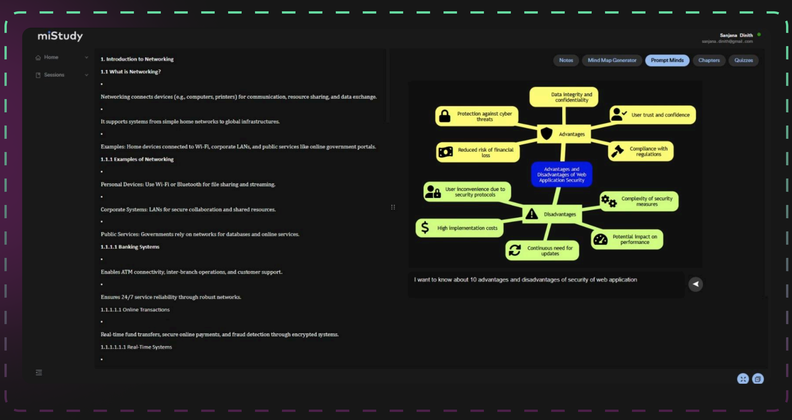
Next Steps

Integrate Alternative Tools.
Evaluate tools like LangChain or Haystack for hierarchical content mapping.
Use local processing tools for pre-structuring the data before sending it to OpenAI.
Optimise OpenAI Usage

Prompt-Based Mind Map Generation

A system where users input a specific topic or question, and the tool generates a hierarchical mind map to explain the concept.

"What are the advantages and disadvantages of application security?"



Progress

Whats done.

PDF Content Extraction

Initial Mind Map Generation

Developed a basic prompt-based mind map generation system

Whats ToDo.

Find a method to convert large documents

When user clicks on one node navigate to the document which that part extract from

Enable navigation through nodes

Implement user-specific storage

Optimise resource linking

IT21249266 – PEIRIS ARD

AI Document Assistant

Facilitate Self Directed Learning

Communicate with documents

Target Problem

- **Lack of personal study support – Outside the lecture hall or classroom, students often receive little to no assistance in their personal studies, leaving them struggling to understand or clarify concepts on their own. Traditional methods fail to provide the interactive guidance needed, leading to inefficiencies in learning and comprehension.**

TABLE 1. Overall, how well did you understand this lecture? (%)

	NESB	ESB
Understood very well	9	34
Understood fairly well	68	58
Did not understand a lot	22	8

Notes: this question not asked in first questionnaire administered (Unit 1).
NESB $N = 174$; ESB $N = 216$.
Differences between groups significant at 0.001 level of confidence.

Proposed Solution

- Introduce an context specific and moderated AI-powered Document Assistant that acts as a virtual lecturer, providing personalized, context-aware support to students during their studies. Utilising RAG architecture together with an LLM and extracting content from PDFs and enabling interactive, conversational queries, the system helps students gain clarity and improve their understanding, bridging the gap between classroom learning and self-study.

Why not a Small Language Model?

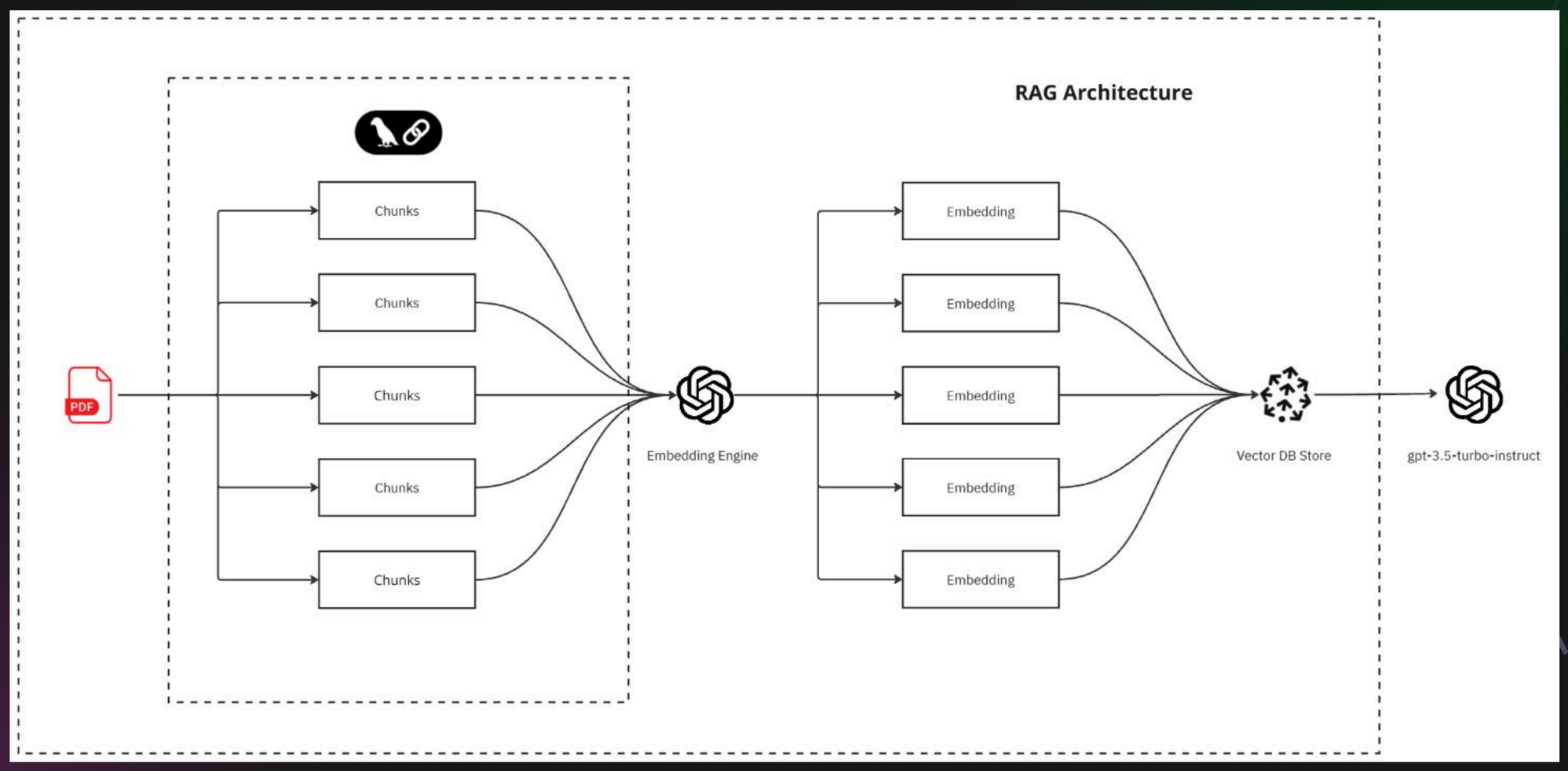
- SLM Lack Depth in Emotional Context
- SLM Limitation on Parameters
- Student-centric Use Case
- Requires significant domain-specific fine tuning
- Less likely to pick up on emotional cues and provides a more generic response

Why OpenAI?

- Has its own embedding model
- Project Requirments
- Out of the box fine-tuned to provide nuanced, context aware and empathetic responses (all LLMs)
- Extensively tested and deployed on wide range of applications
- Comprehensive documentation with Langchain
- Research and Education Focused

	Claude 3 Opus	Claude 3 Sonnet	Claude 3 Haiku	GPT-4	GPT-3.5	Gemini 1.0 Ultra	Gemini 1.0 Pro
Undergraduate level knowledge *MMLU*	86.8% 5 shot	79.0% 5-shot	75.2% 5-shot	86.4% 5-shot	70.0% 5-shot	83.7% 5-shot	71.8% 5-shot
Graduate level reasoning *GPQA, Diamond*	50.4% 0-shot CoT	40.4% 0-shot CoT	33.3% 0-shot CoT	35.7% 0-shot CoT	28.1% 0-shot CoT	–	–
Grade school math *GSM8K*	95.0% 0-shot CoT	92.3% 0-shot CoT	88.9% 0-shot CoT	92.0% 5-shot CoT	57.1% 5-shot	94.4% Maj1@32	86.5% Maj1@32
Math problem-solving *MATH*	60.1% 0-shot CoT	43.1% 0-shot CoT	38.9% 0-shot CoT	52.9% 4-shot	34.1% 4-shot	53.2% 4-shot	32.6% 4-shot
Multilingual math *MGSM*	90.7% 0-shot	83.5% 0-shot	75.1% 0-shot	74.5% 8-shot	–	79.0% 8-shot	63.5% 8-shot
Code *HumanEval*	84.9% 0-shot	73.0% 0-shot	75.9% 0-shot	67.0% 0-shot	48.1% 0-shot	74.4% 0-shot	67.7% 0-shot
Reasoning over text *DROP, F1 score*	83.1 3-shot	78.9 3-shot	78.4 3-shot	80.9 3-shot	64.1 3-shot	82.4 Variable shots	74.1 Variable shots
Mixed evaluations *BIG-Bench-Hard*	86.8% 3-shot CoT	82.9% 3-shot CoT	73.7% 3-shot CoT	83.1% 3-shot CoT	66.6% 3-shot CoT	83.6% 3-shot CoT	75.0% 3-shot CoT
Knowledge Q&A *ARC-Challenge*	96.4% 25-shot	93.2% 25-shot	89.2% 25-shot	96.3% 25-shot	85.2% 25-shot	–	–
Common Knowledge *HellaSwag*	95.4% 10-shot	89.0% 10-shot	85.9% 10-shot	95.3% 10-shot	85.5% 10-shot	87.8% 10-shot	84.7% 10-shot

Architecture



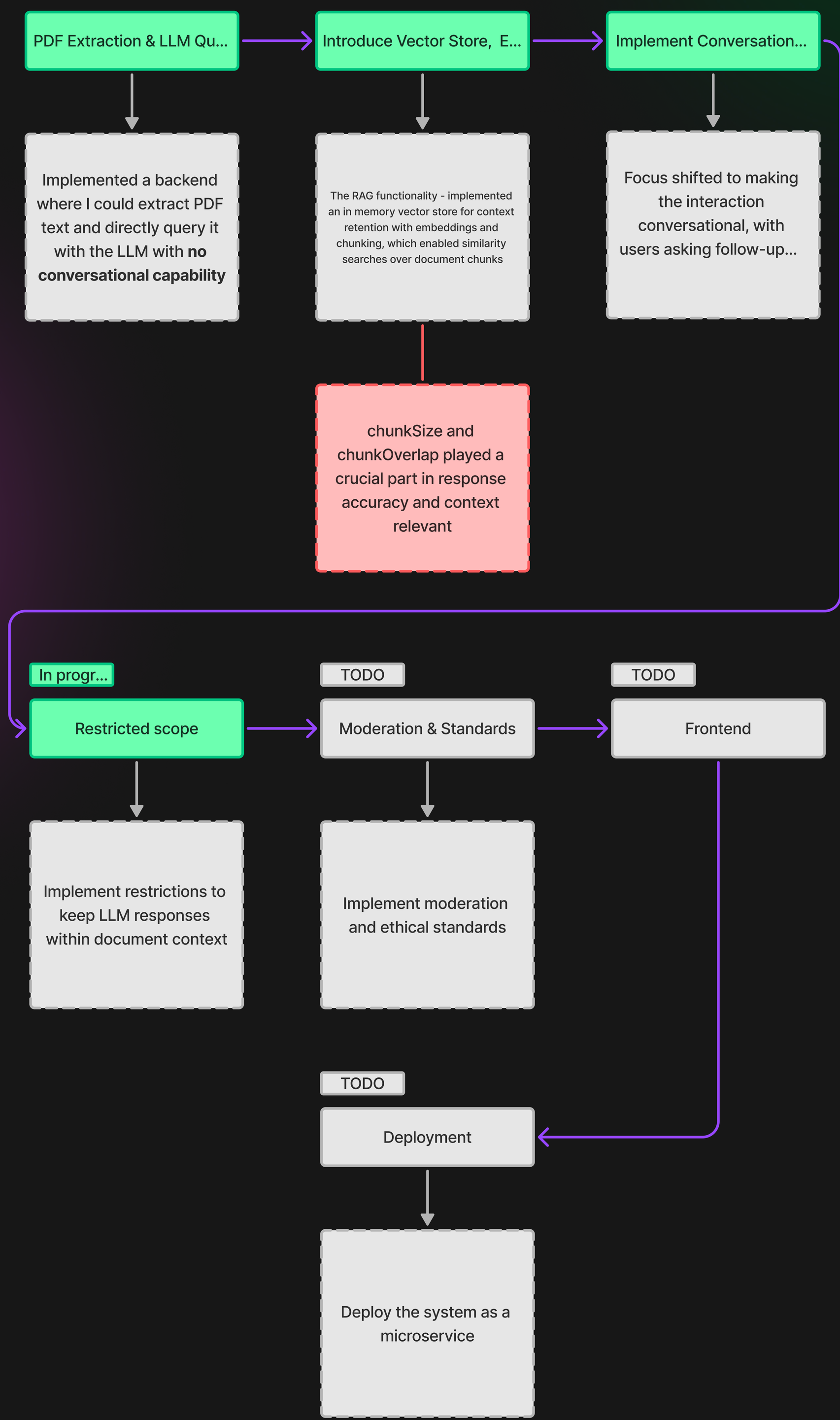
Why RAG is the best solution?

- Common problems in Language Models are lack of domain knowledge, hallucinations and data cut-offs
- RAG provides domain specific data, it significantly reduces hallucination
- No extensive fine tuning or model training necessary
- Cost efficient

Langchain

- A framework for developing apps powered by LLMs
- Connects LLMs to the sources
- Support for Javascript, Typescript & Python
- Facilitates In-Memory Vector storage

Progress so far (60%)



Proof of Work

```typescript
import { OpenAI } from "@langchain/openai";
import dotenv from "dotenv";

const env = process.env.NODE_ENV || "local";
dotenv.config({ path: `.env.${env}` });

const openAIModel = new OpenAI({
  openAIApiKey: process.env.OPENAI_API_KEY!,
  model: process.env.OPENAI_MODEL!,
});

export const queryLLM = async (text: string): Promise<string> => {
  const prompt = `
      ${process.env.OPENAI_PROMPT}
      ${text}
  `;

  const response = await openAIModel.invoke(prompt);
  return response;
};
```

```typescript
import type { Document } from '@langchain/core/documents';
import { OpenAIEmbeddings } from '@langchain/openai';
import dotenv from 'dotenv';
import { RecursiveCharacterTextSplitter } from 'langchain/text_splitter';
import { MemoryVectorStore } from 'langchain/vectorstores/memory';

const env = process.env.NODE_ENV || "local";
dotenv.config({ path: `.env.${env}` });

const embeddings = new OpenAIEmbeddings({
    model: process.env.EMBEDDING_MODEL,
    openAIApiKey: process.env.OPENAI_API_KEY!,
});

let vectorStore: MemoryVectorStore | null = null;

export const initializeVectorStore = async (text: string): Promise<void> => {
    const documents = await createDocumentsFromText(text);

    if (!vectorStore) {
        vectorStore = new MemoryVectorStore(embeddings);
    }

    await vectorStore.addDocuments(documents);
};

const createDocumentsFromText = async (text: string): Promise<Document[]> => {
    const splitter = new RecursiveCharacterTextSplitter({
        chunkSize: 1500, // Increase or decrease this value based on content
        chunkOverlap: 300, // Increase overlap to ensure context is maintained
    });

    const chunks = await splitter.createDocuments([text]);
    return chunks.map(chunk => ({
        pageContent: chunk.pageContent,
        metadata: { source: 'extracted-pdf' },
    }));
};

export const getRelevantContext = async (query: string): Promise<string[]> => {
    if (!vectorStore) throw new Error('Vector store is not initialized');

    const result = await vectorStore.similaritySearch(query, 3);
    return result.map(r => r.pageContent);
};

export const cleanUpVectorStore = (): void => {
        vectorStore = null;
};
```

Proof of Work

POST http://localhost:4001/ask-question Send

Params Authorization Headers (8) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
  "question": "Provide some questions",
    "sessionId": "6d560bc0-9db0-4b6f-9920-6790600d96b8"
}
```


Body Cookies Headers (7) Test Results 200 OK 7.66 s 2.47 KB Save Response

Pretty Raw Preview Visualize JSON

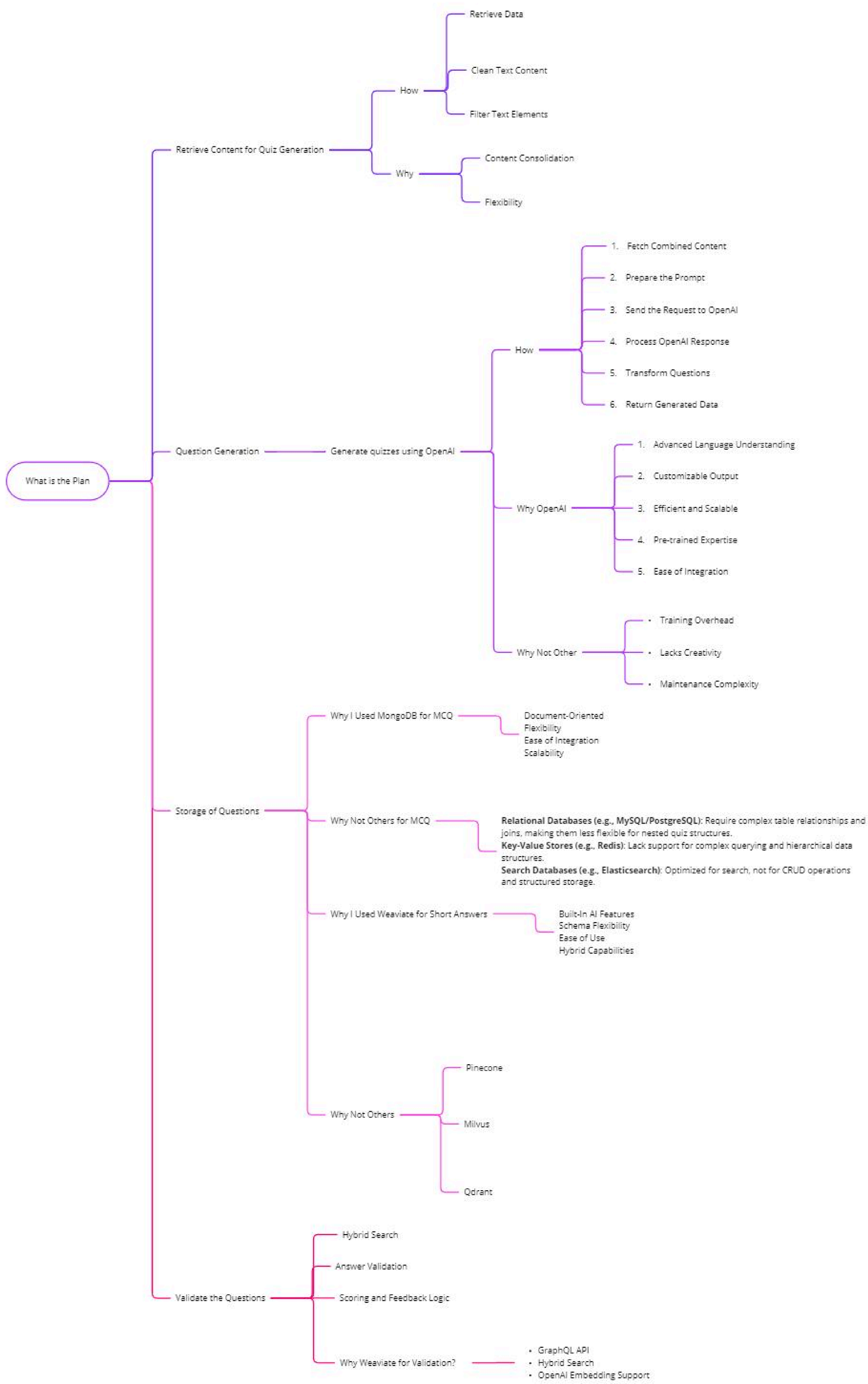

```
{
    "llmResponse": "\n1. Can you explain the roles of service requester and service provider in the context of web services?\n\n2. How does the WSDL (Web Services Description Language) play a part in the overall architecture of web services?\n\n3. What is the purpose of UDDI (Universal Description, Discovery, and Integration) in the context of web services?\n\n4. What is the significance of SOAP (Simple Object Access Protocol) in the web services protocol stack?\n\n5. Can you provide an example of how web services can achieve interoperability between different applications?\n\n6. What are the benefits of using web services, such as platform independence and programming language neutrality?\n\n7. How does the use of standardized protocols in the web services protocol stack benefit businesses?\n\n8. Can you explain the characteristics of web services, such as being XML-based and loosely coupled?\n\n9. How do web services support both synchronous and asynchronous operations?\n\n10. What is the role of RPCs (Remote Procedure Calls) in web services?",
    "chatHistory": [
        {
            "role": "assistant",
            "content": "PDF has been processed. You can ask your questions now."
        },
        {
            "role": "user",
            "content": "Provide some questions"
```

IT21360428 – Monali G.M.N.

Adaptive Quizzes Generation

Adaptive quizzes tailored to individual learners

Current Workflow



Quiz Generation

Current Workflow Overview

- **Automatically create quizzes tailored to the lecture content**

Steps Involved

- **Retrieve content from the pdf extraction**
- **Clean and preprocess the extracted text.**
- **Use OpenAI model to:**
  - **Generate multiple-choice questions (MCQs) ans ShortAnswer questions based on content.**
  - **Define quiz difficulty levels (0-100 scale for easy to complex).**

```typescript
try {
        // Fetch the extracted elements from the new endpoint
        const response = await axios.get(`http://localhost:${process.env.MINDMAP_PORT}/lms/pdfcontentExtractElements`);

        // Ensure response data is valid
        const extractedElements = response.data;
        if (!extractedElements || extractedElements.length === 0) {
            return res.status(400).json({ message: 'No valid content data found.' });
        }

        // Combine the text of all extracted elements
        const combinedContent = extractedElements
            .map(element => element.text.trim()) // Extract text only
            .filter(text => text) // Exclude empty or undefined text
            .join('\n'); // Combine into a single string

        // Generate quiz questions using OpenAI model
        const { title, questions } = await generateQuiz({ content: combinedContent }, resolvedDifficultyLevel);

        // Ensure questions are generated
        if (!questions || questions.length === 0) {
            return res.status(400).json({ message: 'No questions could be generated from the content.' });
        }

        // Create a new Quiz document
        const quiz = new Quiz({
            title,
            // contentId,
            userID,
            difficultyLevel: resolvedDifficultyLevel,
            questions,
        });

        // Save the quiz to the database
        const savedQuiz = await quiz.save();

        // Return success response with the quiz
        // res.status(201).json({ message: 'Quiz created successfully.', quiz: questions });
        res.status(201).json({
            message: 'Quiz created successfully.',
            quiz: {
                id: savedQuiz._id,
                title: savedQuiz.title,
                difficultyLevel: savedQuiz.difficultyLevel,
                questions: savedQuiz.questions,
            },
        });
    }
```

```typescript
try {
    // Fetch the extracted elements from the new endpoint
    const response = await axios.get(`http://localhost:${process.env.MINDMAP_PORT}/lms/pdfcontentExtractElements`);

    // Ensure response data is valid
    const extractedElements = response.data;
    if (!extractedElements || extractedElements.length === 0) {
        return res.status(400).json({ message: 'No valid content data found.' });
    }

    // Combine the text of all extracted elements
    const combinedContent = extractedElements
    .map(element => element.text.trim()) // Extract text only
    .filter(text => text) // Exclude empty or undefined text
    .join('\n'); // Combine into a single string

    // Generate quiz questions using OpenAI model
    const { title, questions } = await generateShortQuiz({ content: combinedContent }, resolvedDifficultyLevel);

    if (!questions || questions.length === 0) {
        return res.status(400).json({ message: 'No questions could be generated from the content.' });
    }

    // Store quiz in Weaviate
    const savedQuiz = await storeQuizInWeaviate({
      title,
      // contentId,
      // userID,
      difficultyLevel:resolvedDifficultyLevel,
      question: questions.map(q => q.question),
      correctAnswer: questions.map(q => q.correctAnswer)
    });

    // Extract quizId from the stored quiz response
    const quizId = savedQuiz.quiz.id;

    res.status(201).json({
      message: 'Quiz created successfully.',
      quiz: {
        id: quizId,
        title: title,
        questions: questions
      }
    });
}
```

```typescript
// Function to generate quiz questions based on content
export const generateQuiz = async (content, difficultyLevel) => {
    const prompt =
        You are a professional quiz generator. Based on the following content, do the following:

        1. Provide a relevant title for the quiz based on the content.
        2. Create 10 multiple-choice quiz questions.
        Each question should have one correct answer and 3 incorrect options. Provide the questions in valid JSON format:

        {
            "title": "Quiz Title Here",  // Add a relevant title here
            "difficultyLevel": ${difficultyLevel},
            "quiz_questions": [
                {
                    "Question": "Your question here",
                    "Correct Answer": "Correct answer here",
                    "Other Answers": ["Incorrect 1", "Incorrect 2", "Incorrect 3"]
                }
            ]
        }

        The difficulty level MUST be exactly ${difficultyLevel}:
        - If the difficulty level is closer to 0, the questions should be easier and more straightforward.
        - If the difficulty level is closer to 100, the questions should be more complex, requiring deeper knowledge and critic
al thinking.

        Content:
        ${JSON.stringify(content)}
    `;

    try {
        const response = await openai.chat.completions.create({
            model: 'gpt-3.5-turbo',
            messages: [{ role: 'user', content: prompt }],
            max_tokens: 3000,
            temperature: 1.0,
        });

        // Extract and return only the quiz text from the response
        const quizText = response.choices[0].message.content.trim();

        // Log the raw response to check for anomalies
        console.log('Raw Response from OpenAI:', quizText);

        // Manually clean the response if necessary
        let cleanQuizText = quizText.replace(/^[\s*]/g, '', {});

        console.log('Cleaned Response from OpenAI:', cleanQuizText);

        // Validate and parse JSON
        let quizData;
        try {
            quizData = JSON.parse(cleanQuizText);
            console.log('Raw Quiz Data from OpenAI:', JSON.stringify(quizData, null, 2));
        } catch (jsonError) {
            console.error('Invalid JSON from OpenAI:', cleanQuizText);
            throw new Error('OpenAI response was not valid JSON.');
        }

        // Validate structure
        if (!quizData || !quizData.quiz_questions || quizData.quiz_questions.length === 0) {
            throw new Error('Invalid or incomplete quiz data.');
        }

        const transformedQuestions = quizData.quiz_questions.map((q) => ({
            question: q.Question,
            correctAnswer: q['Correct Answer'],
            otherAnswers: q['Other Answers'],
        }));

        // Extract the title
        const title = quizData.title || 'Untitled Quiz';

        return { title, questions: transformedQuestions };
    }
```

```typescript
// Function to generate quiz questions based on content
export const generateShortQuiz = async (content, difficultyLevel) => {
    const prompt =
        You are a professional quiz generator. Based on the following content, do the following:

        1. Provide a relevant title for the quiz based on the content.
        2. Create 10 short-answer quiz questions in valid JSON format.
        Each question must have one correct answer.

        The JSON format should look like this:
        {
            "title": "Quiz Title Here",   // Add a relevant title here
            "difficultyLevel": ${difficultyLevel},
            "quiz_questions": [
                {
                    "question": "Your question here",
                    "correctAnswer": "Correct answer here"
                },
                ...
            ]
        }

        The difficulty level MUST be exactly ${difficultyLevel}:
        - If the difficulty level is closer to 0, the questions should be easier and more straightforward.
        - If the difficulty level is closer to 100, the questions should be more complex, requiring deeper knowledge and critic
al thinking.

        Content:
        ${JSON.stringify(content)}
    `;

    try {
        const response = await openai.chat.completions.create({
            model: 'gpt-3.5-turbo',
            messages: [{ role: 'user', content: prompt }],
            max_tokens: 3000,
            temperature: 0.7,
        });

        // Extract and return only the quiz text from the response
        const quizText = response.choices[0].message.content.trim();
        console.log('Raw Quiz Data from OpenAI:', quizText);

        // Manually clean the response if necessary
        let cleanQuizText = quizText.replace(/^[\s*]/g, '', {});
        console.log('Cleaned Response from OpenAI:', cleanQuizText);

        // Validate and parse JSON
        const quizData = JSON.parse(cleanQuizText);
        console.log('Parsed OpenAI JSON:', JSON.stringify(quizData, null, 2));

        const transformedQuestions = quizData.quiz_questions.map((q) => ({
            question: q.question,
            correctAnswer: q.correctAnswer,
        }));

        console.log('Transformed Questions:', transformedQuestions);

        // Extract the title
        const title = quizData.title || 'Untitled Quiz';

        return { title, questions: transformedQuestions };
    }
```

Why OpenAI ?

- **Advanced Language Understanding**
- **Customizable Output**
- **Efficient and Scalable**
- **Pre-trained Expertise**
- **Ease of Integration**

Why Not Other ?

- **Lacks Creativity**
- **Maintenance Complexity**
- **Training Overhead**

Saving Quizzes into Database.

Storage Mechanisms

MongoDB (for MCQs)

- Relational Structure:
  - MCQs have a predictable, structured format: question, correct answer, and other answer options.
  - MongoDB's schema-less nature works well for such structured yet flexible formats.
- Advantages for MCQs:
  - Supports indexing for faster querying of large datasets.
  - Suitable for traditional question-answer relationships.
  - Scales horizontally, making it apt for systems expecting many users simultaneously generating quizzes.

Database Structure


```
_id : ObjectId('6750a3aaf5247cd54998c51e')
title : "Software Engineering Aspects Quiz"
difficultyLevel : 50
questions : Array (10)
  0: Object
    question : "What is the primary focus of software engineering?"
    correctAnswer : "Design, development, testing, and maintenance of software applications"
    otherAnswers : Array (3)
      0: "Hardware design"
      1: "Data analysis"
      2: "Network security"
    _id : ObjectId('6750a3aaf5247cd54998c51f')
  1: Object
    question : "What is a key step in Requirements Analysis?"
    correctAnswer : "Eliciting requirements from stakeholders"
    otherAnswers : Array (3)
      0: "Writing code immediately"
      1: "Skipping validation"
      2: "Ignoring non-functional requirements"
    _id : ObjectId('6750a3aaf5247cd54998c520')
  2: Object
    question : "What does System Design phase define?"
    correctAnswer : "Architecture and structure of the application"
    otherAnswers : Array (3)
      0: "User interface only"
      1: "Marketing strategy"
      2: "Customer support process"
    _id : ObjectId('6750a3aaf5247cd54998c521')
  3: Object
    question : "Which practice is important during Implementation phase?"
```


Weaviate (for Short-Answer Questions)

- Semantic Search with Vector Embeddings:
  - Short-answer questions require understanding the meaning of text rather than exact matches.
  - Weaviate uses text embeddings (vector representations of text) to enable semantic similarity searches.
- Hybrid Search Capabilities:
  - Combines keyword-based and AI-driven semantic searches to evaluate user input more effectively.
- Integration with AI Models:
  - Built-in support for OpenAI's models ensures seamless alignment with your quiz generation and validation pipeline.

Database Structure


```
{
  "data": {
    "Get": {
      "Quiz": [
        {
          "_additional": {
            "id": "05d77089-5716-462f-8a20-c9ea9cc28319"
          },
          "correctAnswer": [
            "To connect devices for communication, resource sharing, and data exchange.",
            "Online government portals.",
            "Reduces costs and improves efficiency.",
            "Local Area Network; used in limited areas like homes, offices, or schools.",
            "Routers.",
            "Physical is the actual arrangement of hardware, while logical defines data flow.",
            "All devices connect to a central hub, providing centralized control and scalability.",
            "Failure of individual devices doesn't disrupt the network.",
            "It affects network speed, redundancy, and troubleshooting efficiency.",
            "WAN connects multiple LANs across large geographic areas, while LAN is limited to a specific area."
          ],
          "difficultyLevel": 70,
          "question": [
            "What is the purpose of networking?",
            "Give an example of a public service that relies on networking.",
            "What is the key benefit of resource sharing in networking?",
            "What is a LAN and where is it commonly used?",
            "What is the main component that connects LANs to external networks like the internet?",
            "What is the difference between physical and logical network topology?",
            "Describe the Star Topology.",
            "What advantage does the Star Topology offer in terms of network disruption?",
            "What is the impact of network topology on troubleshooting efficiency?",
            "How does WAN differ from LAN in terms of geographical coverage?"
          ],
          "title": "Networking Basics Quiz"
        },
```


Why the Dual-Database Approach?

By combining MongoDB and Weaviate, you leverage the strengths of both systems:

- MongoDB Strengths:
  - Handles structured data like MCQs efficiently.
  - Simple query mechanisms for fixed formats.
  - Cost-effective for basic storage needs.
- Weaviate Strengths:
  - Excels at unstructured data and complex text comparisons.
  - Enables advanced validation for short-answer responses using semantic understanding.
  - Enhances accuracy and adaptability in grading responses with partial correctness.

Comparison With Alternatives

- Why not other vector databases (e.g., Pinecone, Milvus)?
  - Weaviate's modular architecture supports both vector search and OpenAI integration.
  - Many alternatives focus only on vector search, lacking hybrid keyword and semantic capabilities.
- Why not solely MongoDB?
  - MongoDB struggles with tasks requiring semantic understanding or similarity scoring.
  - Purely relational databases cannot perform hybrid search or handle embeddings effectively.

Validating Answers Effectively

Ensure that user answers are accurately evaluated with a balance of precision and fairness, accommodating both MCQs and short-answer questions.

MCQ Validation

How It Works

- Compare the user's selected option directly with the correct answer stored in MongoDB.
- MongoDB's schema structure allows efficient and straightforward lookups.

Key Features

- Immediate feedback:
  - Identifies correct or incorrect answers without delay.
- Scalable queries for high volumes of users simultaneously.

Technical Tools Used

- MongoDB's Query API:
  - Simple queries for retrieving questions and correct answers.
  - Performance optimized with indexing on quiz fields.

Short-Answer Validation

How It Works

- User answers are validated against stored correct answers using semantic search and exact matching.
- Weaviate's Hybrid Search combines:
  - Keyword-based search for direct matches.
  - Semantic similarity to evaluate understanding even if phrased differently.

Key Features

- Exact Match Checks:
  - Ensures that verbatim answers are flagged as correct.
- Semantic Similarity Scoring:
  - Uses embeddings from OpenAI's text models to analyze the meaning of user responses.
  - Enables partial correctness scoring for answers that are close but not exact.
- Detailed Feedback:
  - Correct: Confirms the user's understanding.
  - Partially Correct: Highlights areas of improvement with suggestions.
  - Incorrect: Explains the correct answer and its context.

Technical Tools Used

- Weaviate GraphQL API:
  - Fetch stored questions and correct answers.
  - Perform hybrid queries blending keyword and vector-based searches.
- OpenAI Embedding Models (e.g., Ada):
  - Generate vector representations of correct answers.
  - Compare these with user input for semantic similarity.

```typescript
try {
    // Fetch the quiz from Weaviate by ID
    const quizResponse = await client.graphql
      .get()
      .withClassName('Quiz')
      .withFields(`question correctAnswer`)
      .withWhere({
        path: ['id'],
        operator: 'Equal',
        valueString: quizId,
      })
      .do();

    if (!quizResponse || !quizResponse.data || quizResponse.data.Get.Quiz.length === 0) {
      return res.status(404).json({ message: 'Quiz not found.' });
    }

    const quiz = quizResponse.data.Get.Quiz[0];

    // Validate each user answer using a hybrid query
    const results = await Promise.all(
      answers.map(async (userAnswer, index) => {
        const { question, answer } = userAnswer;

        // Perform hybrid search for the correct answer
        const searchResponse = await client.graphql
          .get()
          .withClassName('Quiz')
          .withFields(`correctAnswer _additional { score }`)
          .withHybrid({
            query: answer, // User's answer
            alpha: 0.75,   // Adjust for keyword vs semantic weighting
          })
          .withWhere({
            path: ['question'],
            operator: 'Equal',
            valueString: question,
          })
          .do();

        // Fetch the correct answer for the question
        const correctAnswer = quiz.correctAnswer[index];

        // Normalize answers to lowercase for a case-insensitive comparison
        const normalizedUserAnswer = answer.trim().toLowerCase();
        const normalizedCorrectAnswer = correctAnswer.trim().toLowerCase();

        let feedback = '';
        let isCorrect = false;

        // Check for exact match
        if (normalizedUserAnswer === normalizedCorrectAnswer) {
          feedback = "Correct!";
          isCorrect = true;
        } else {
          // Calculate hybrid score for the answer
          const hybridScore = searchResponse.data.Get.Quiz[0]._additional.score;

          // Adjusting logic based on hybrid score
          if (hybridScore > 0.9) {
            feedback = "Correct!";
            isCorrect = true;
          } else if (hybridScore > 0.75) {
            feedback = `Partially correct. Review the material for a better understanding. The correct answer is: ${
correctAnswer}.`;
          } else {
            feedback = `Incorrect. The correct answer is: ${correctAnswer}`;
          }
        }

        return {
          question,
          userAnswer: answer,
          hybridScore: searchResponse.data.Get.Quiz[0]._additional.score,
          isCorrect,
          feedback,
        };
      }))
    );
```

Why This Approach?

Precision for MCQs

- Simple structure and exact matching make MongoDB the ideal solution for traditional question-answer formats.

Flexibility for Short Answers

- Weaviate's vector-based approach ensures nuanced grading for open-ended responses.
- Hybrid search allows balanced evaluations combining semantic and keyword relevance.

IT21360428 - Monali G.M.N.

Progress

Whats done.

- Quiz Generation
- Saving Quizzes into Database
- Validating Answers Effectively

Whats ToDo.

- Personalized Quizzes
- Personalized Feedback
- Enhanced Accuracy